ZKFault: Fault Attack Analysis on Zero-Knowledge Based Post-Quantum Digital Signature Schemes

Puja Mondal, Supriya Adhikary, Suparna Kundu, and Angshuman Karmakar

Asiacrypt 2024, Kolkata, India

NIST's Additional Digital Signature

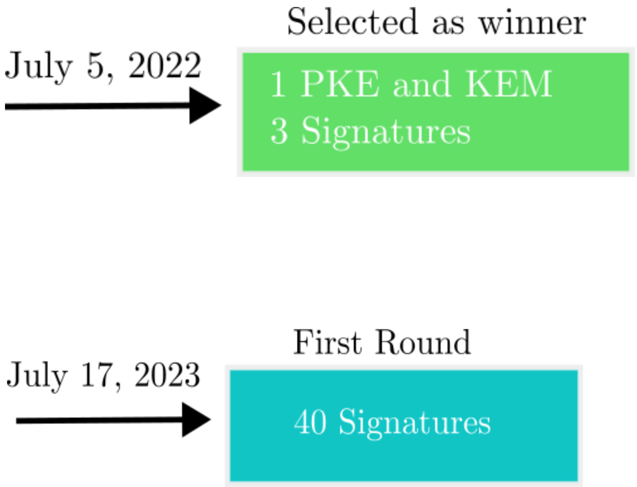
July 5, 2022

Selected as winner

1 PKE and KEM
3 Signatures

PKE and KEM	Signatures
CRYSTALS-Kyber	CRYSTALS-Dilithium FALCON SPHINCS+

NIST's Additional Digital Signature



July 5, 2022

Selected as winner

1 PKE and KEM
3 Signatures

July 17, 2023

First Round

40 Signatures

PKE and KEM	Signatures
CRYSTALS-Kyber	CRYSTALS-Dilithium
	FALCON
	SPHINCS+

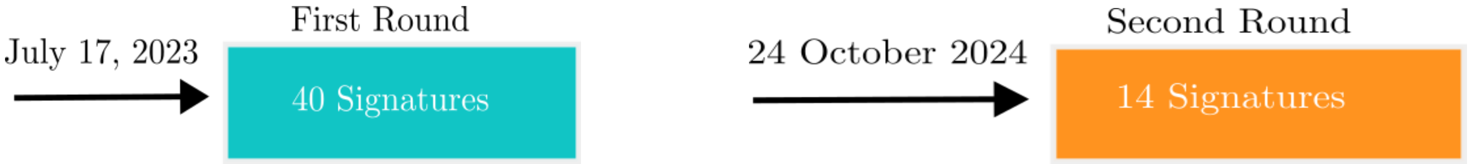
NIST's Additional Digital Signature

Selected as winner

July 5, 2022

1 PKE and KEM
3 Signatures

PKE and KEM	Signatures
CRYSTALS-Kyber	CRYSTALS-Dilithium FALCON SPHINCS+



First Round

July 17, 2023

40 Signatures

Second Round

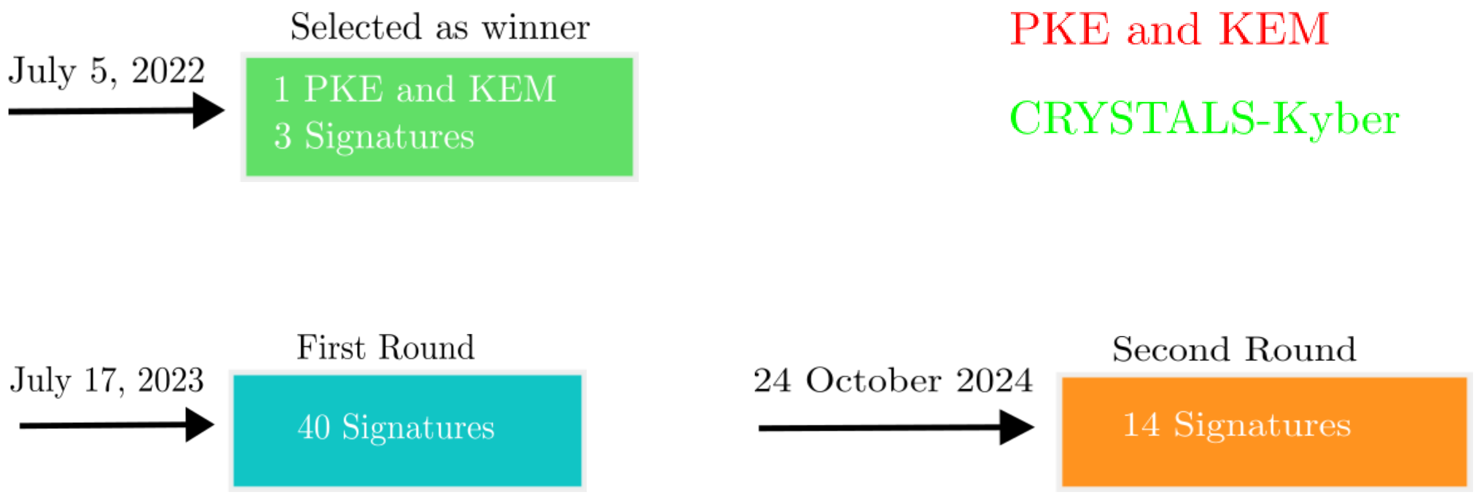
24 October 2024

14 Signatures

CROSS **FEAST** **HAWK** **LESS** **MAYO** **Mirath** **MQOM**

PERK **QR-UOV** **RYDE** **SDitH** **SNOVA** **SQIsign** **UOV**

NIST's Additional Digital Signature



July 5, 2022

Selected as winner

1 PKE and KEM
3 Signatures

July 17, 2023

First Round

40 Signatures

24 October 2024

Second Round

14 Signatures

PKE and KEM

CRYSTALS-Kyber

Signatures

CRYSTALS-Dilithium
FALCON
SPHINCS+

✓			✓			
CROSS	**FEAST**	**HAWK**	**LESS**	**MAYO**	**Mirath**	**MQOM**
PERK	**QR-UOV**	**RYDE**	**SDitH**	**SNOVA**	**SQIsign**	**UOV**

Interactive-Zero-Knowledge (ZK) Framework

Prover

Verifier

Interactive-Zero-Knowledge (ZK) Framework

Prover

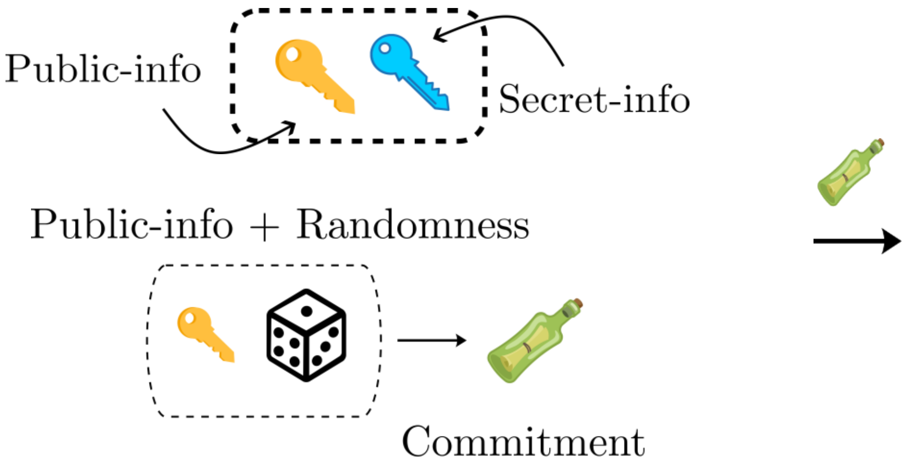
Verifier

Public-info
Secret-info

Interactive-Zero-Knowledge (ZK) Framework

Prover

Verifier



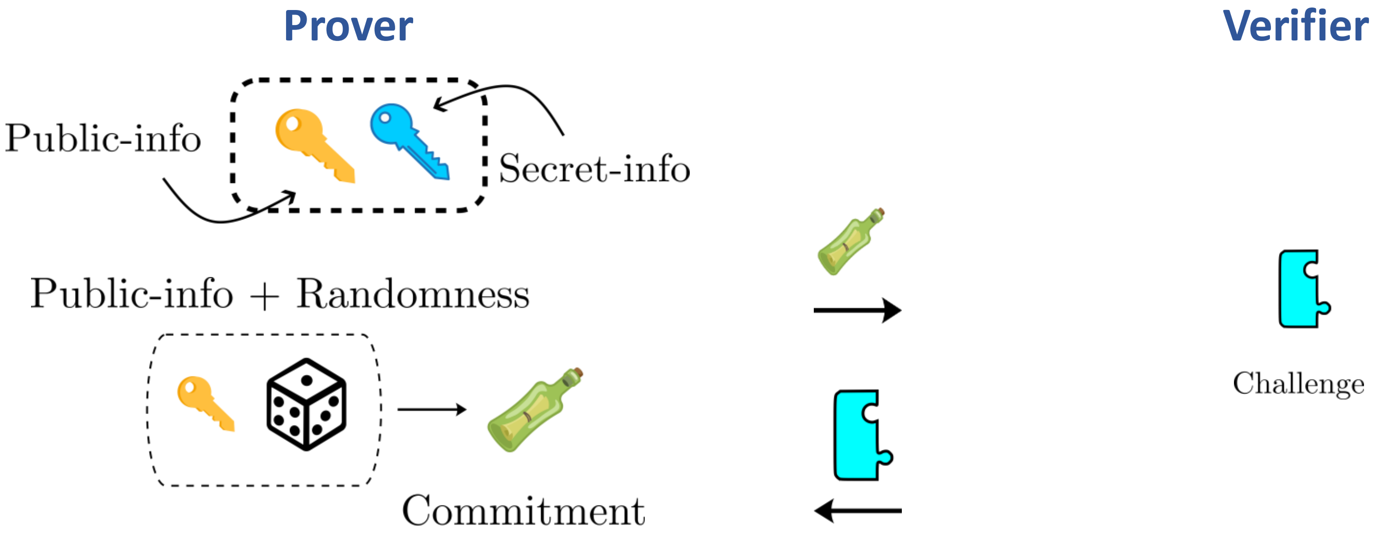
Public-info

Secret-info

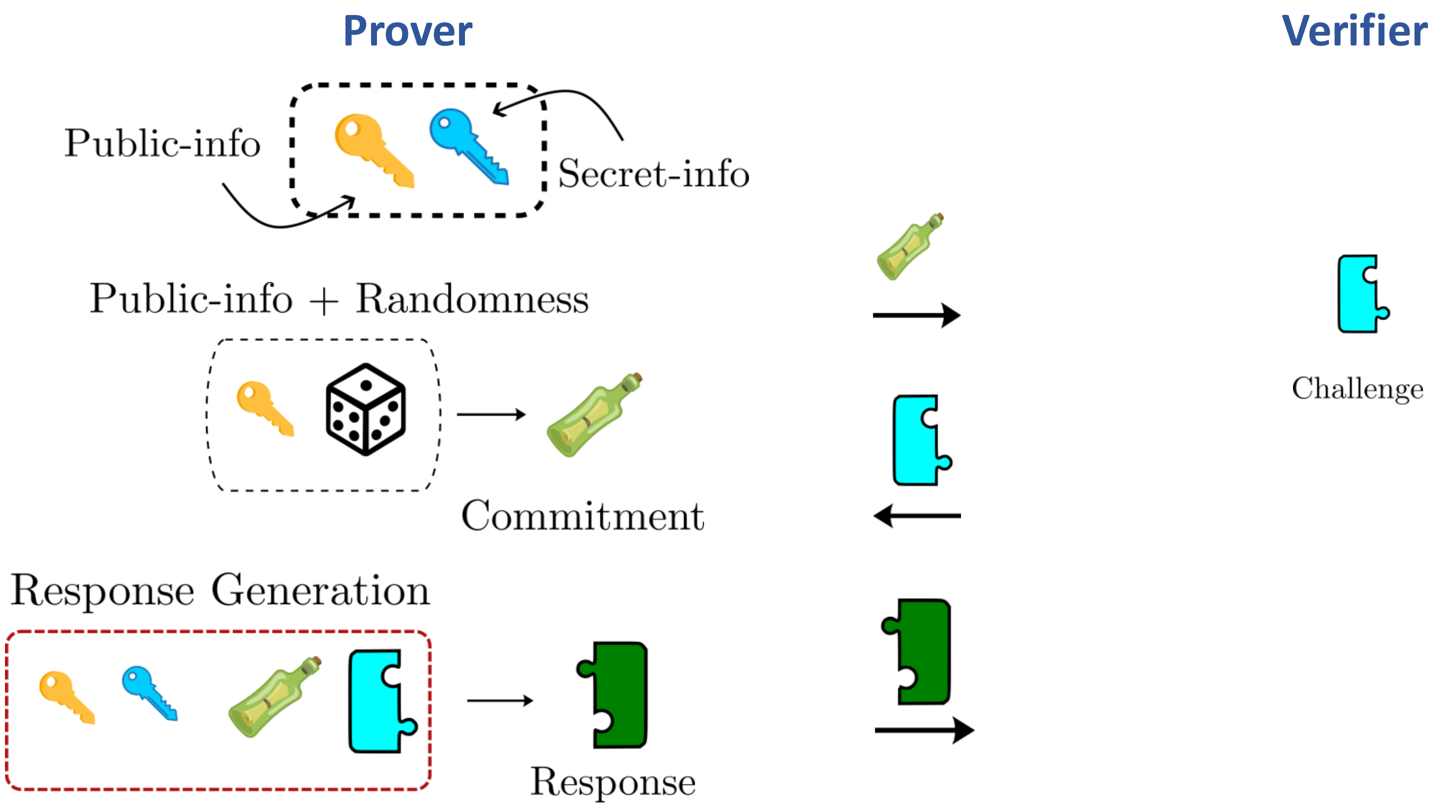
Public-info + Randomness

Commitment

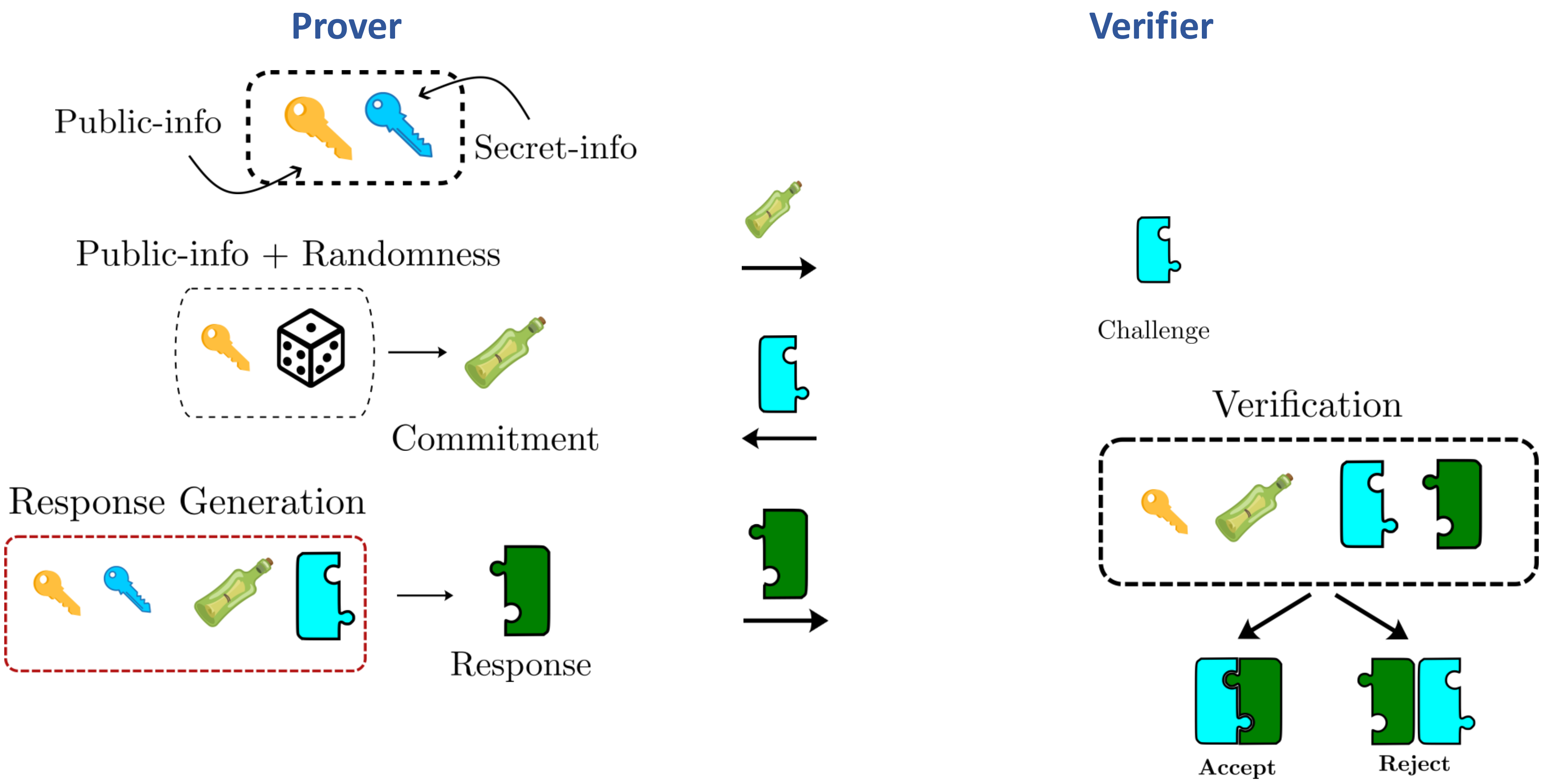
Interactive-Zero-Knowledge (ZK) Framework



Interactive-Zero-Knowledge (ZK) Framework



Interactive-Zero-Knowledge (ZK) Framework



Prover

Verifier

Public-info

Secret-info

Public-info + Randomness

Commitment

Response Generation

Response

Challenge

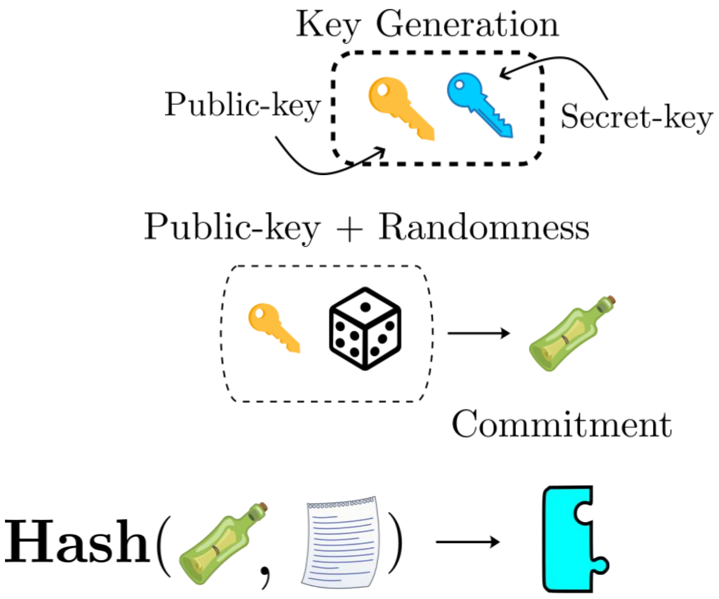
Verification

Accept

Reject

ZK with Fiat-Shamir Transformation

Signer **Verifier**



Key Generation

Public-key

Secret-key

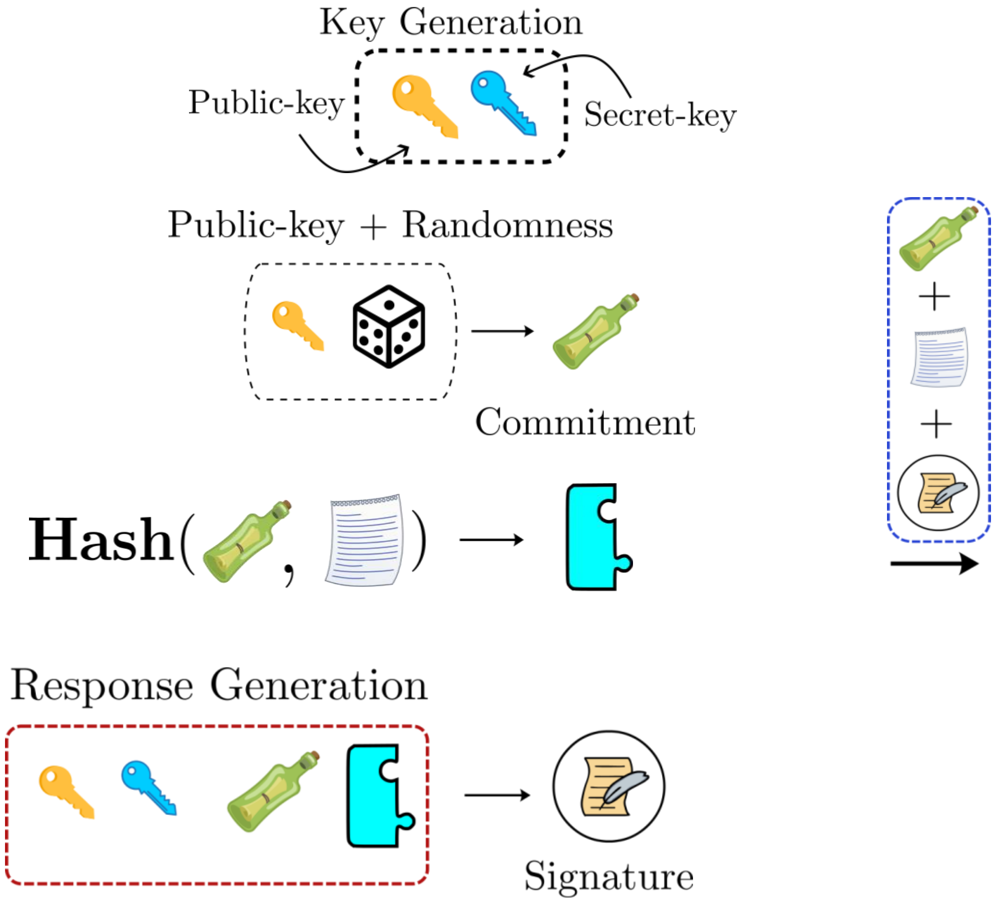
Public-key + Randomness

Commitment

Hash(,) →

ZK with Fiat-Shamir Transformation

Signer **Verifier**



Key Generation

Public-key

Secret-key

Public-key + Randomness

Commitment

Hash(,)

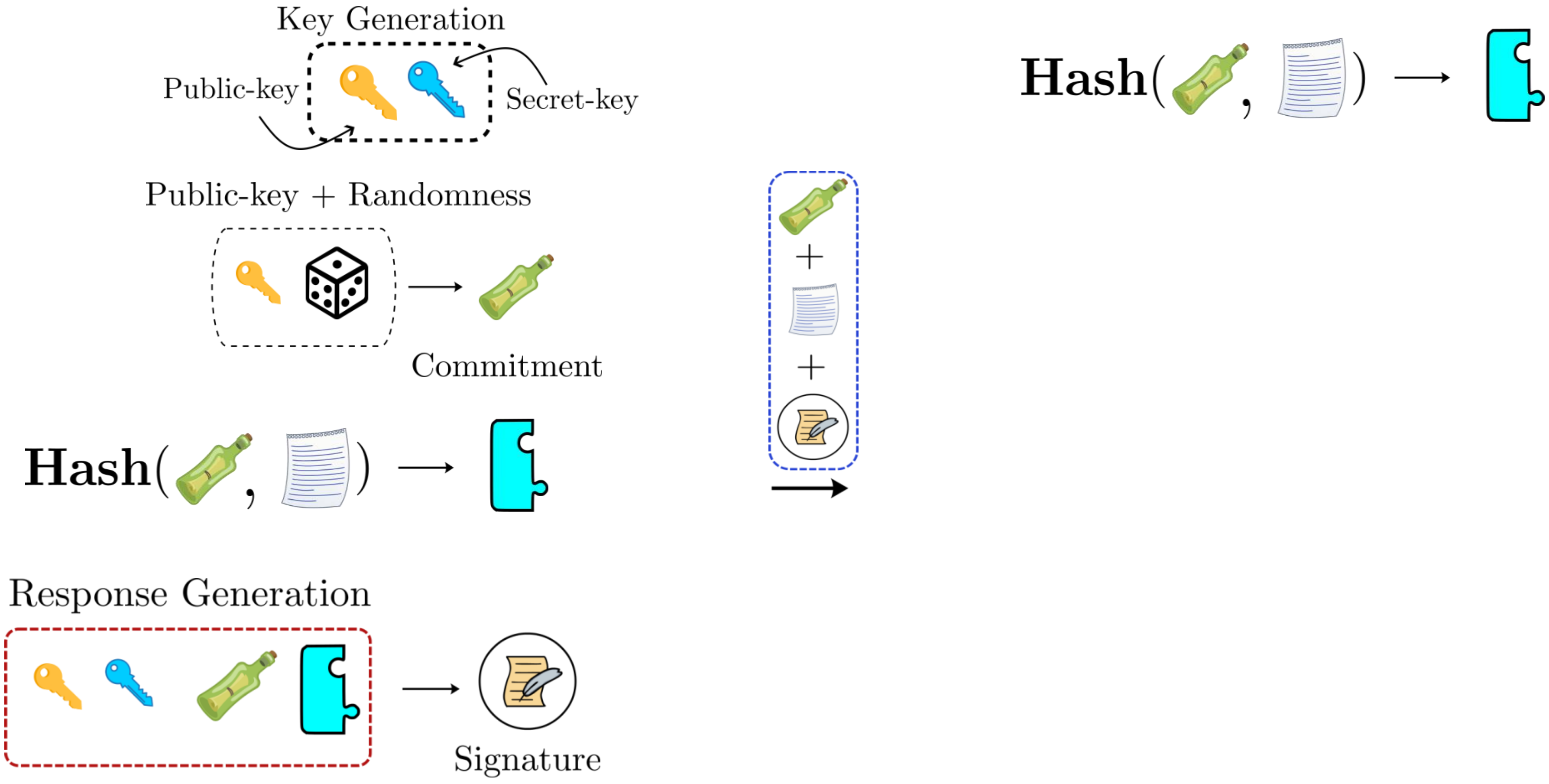
Response Generation

Signature

ZK with Fiat-Shamir Transformation

Signer

Verifier



Key Generation

Public-key

Secret-key

Public-key + Randomness

Commitment

Hash(,) →

+

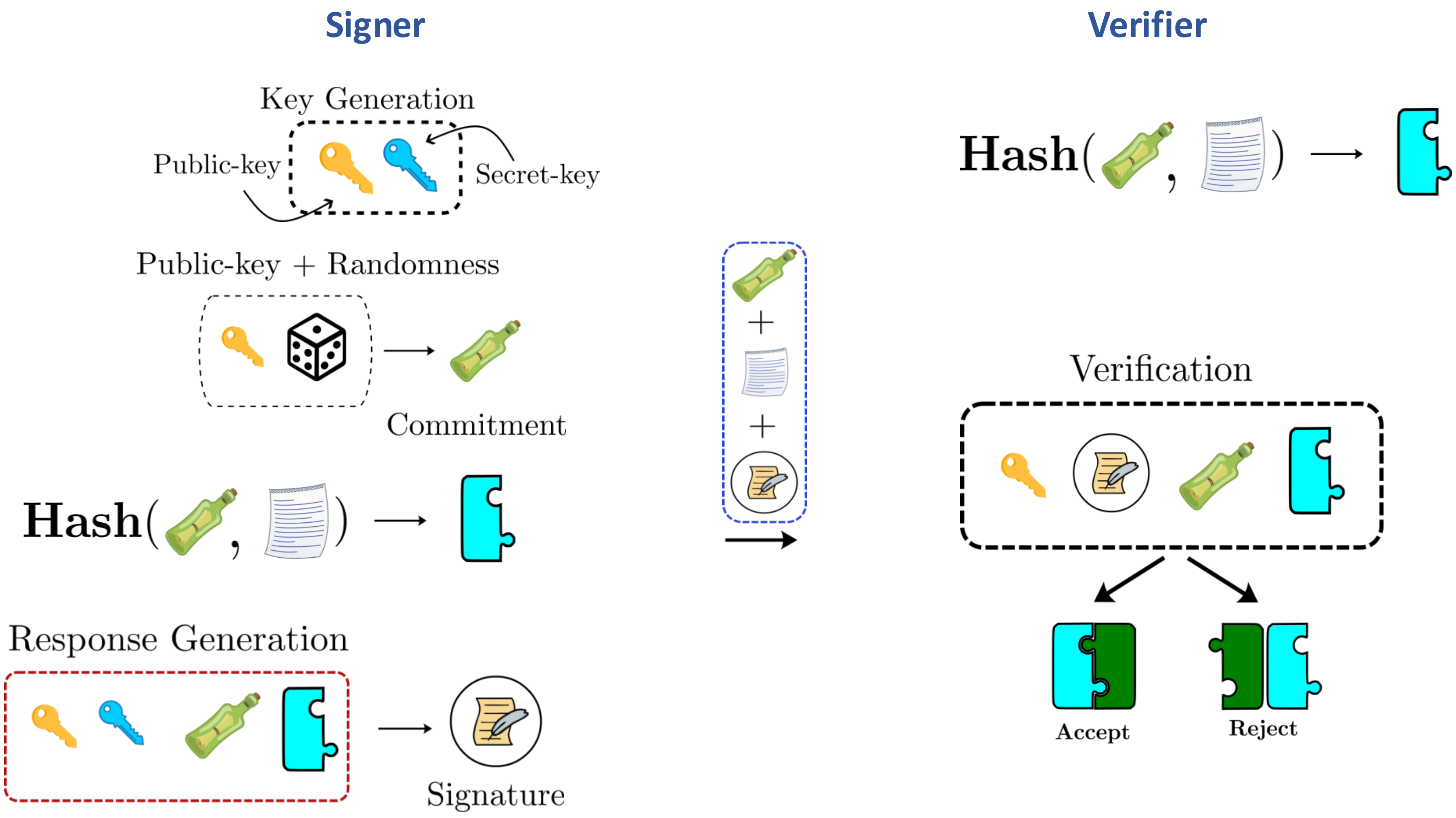
+

Response Generation

Signature

Hash(,) →

ZK with Fiat-Shamir Transformation



LESS

Signer

Verifier

🔑 : $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s$

🔑 : $\boldsymbol{G}_0, \boldsymbol{G}_1 = S(\boldsymbol{G}_0\boldsymbol{Q}_1), \boldsymbol{G}_2, \cdots, \boldsymbol{G}_s$

LESS

Signer

🔑 : $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s$

🔑 : $\boldsymbol{G}_0, \boldsymbol{G}_1 = S(\boldsymbol{G}_0\boldsymbol{Q}_1), \boldsymbol{G}_2, \cdots, \boldsymbol{G}_s$

Sample : $\boldsymbol{Q}'_1, \boldsymbol{Q}'_2, \cdots, \boldsymbol{Q}'_t$

$\boldsymbol{G}'_1 = S(\boldsymbol{G}_0\boldsymbol{Q}'_1), \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t$

Verifier

LESS

Signer

Verifier

🔑 : $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s$

🔑 : $\boldsymbol{G}_0, \boldsymbol{G}_1 = S(\boldsymbol{G}_0\boldsymbol{Q}_1), \boldsymbol{G}_2, \cdots, \boldsymbol{G}_s$

$$\boxed{\begin{array}{l} \textbf{Sample}: \boldsymbol{Q}'_1, \boldsymbol{Q}'_2, \cdots, \boldsymbol{Q}'_t \\ \boldsymbol{G}'_1 = S(\boldsymbol{G}_0\boldsymbol{Q}'_1), \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t \end{array}}$$

🍾 : $\textbf{Hash}(\boldsymbol{G}'_1, \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t)$

Commitment

Challenge

$\textbf{Hash}($ 🍾, 📄 $) \longrightarrow$ 🧩 : $\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_t} \in \{\mathbf{0}, \cdots, \mathbf{s}\}$

LESS

Signer

Verifier

🔑 : $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s$

🔑 : $\boldsymbol{G}_0, \boldsymbol{G}_1 = S(\boldsymbol{G}_0\boldsymbol{Q}_1), \boldsymbol{G}_2, \cdots, \boldsymbol{G}_s$

Sample : $\boldsymbol{Q}'_1, \boldsymbol{Q}'_2, \cdots, \boldsymbol{Q}'_t$

$\boldsymbol{G}'_1 = S(\boldsymbol{G}_0\boldsymbol{Q}'_1), \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t$

🍾 : $\mathbf{Hash}(\boldsymbol{G}'_1, \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t)$

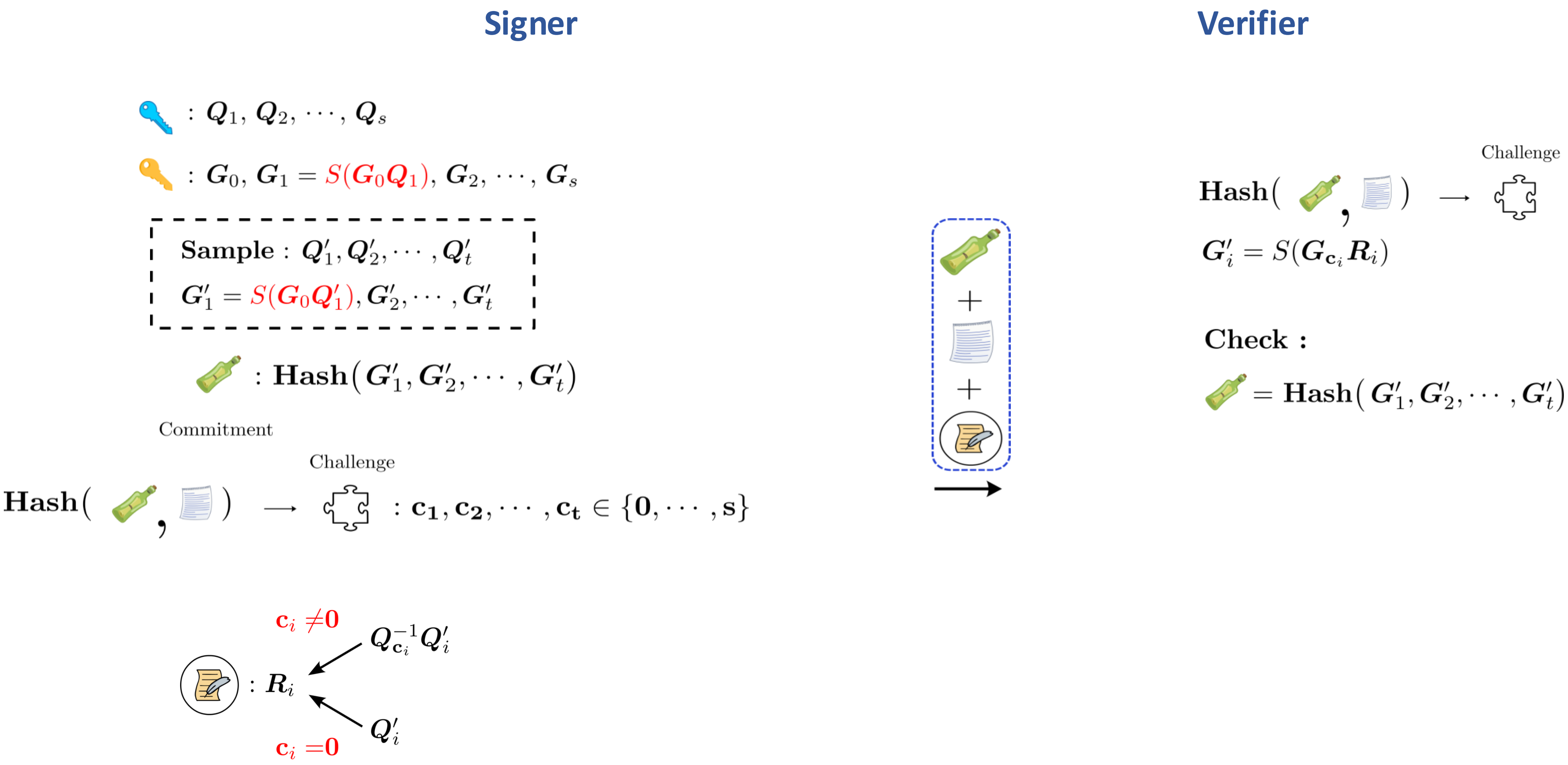
Commitment

Challenge

$\mathbf{Hash}($ 🍾, 📄 $) \longrightarrow$ 🧩 : $\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_t} \in \{\mathbf{0}, \cdots, \mathbf{s}\}$

$$\begin{tikzcd} & \boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i \arrow[dl, "\mathbf{c}_i \neq \mathbf{0}"'] \\ ✍ : \boldsymbol{R}_i & \\ & \boldsymbol{Q}'_i \arrow[ul, "\mathbf{c}_i = \mathbf{0}"] \end{tikzcd}$$

LESS



Signer

🔑 : $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s$

🔑 : $\boldsymbol{G}_0, \boldsymbol{G}_1 = S(\boldsymbol{G}_0\boldsymbol{Q}_1), \boldsymbol{G}_2, \cdots, \boldsymbol{G}_s$

Sample : $\boldsymbol{Q}'_1, \boldsymbol{Q}'_2, \cdots, \boldsymbol{Q}'_t$

$\boldsymbol{G}'_1 = S(\boldsymbol{G}_0\boldsymbol{Q}'_1), \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t$

🍾 : $\mathbf{Hash}(\boldsymbol{G}'_1, \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t)$

Commitment

Challenge

$\mathbf{Hash}($ 🍾, 📄 $) \longrightarrow$ 🧩 : $\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_t} \in \{\mathbf{0}, \cdots, \mathbf{s}\}$

$\mathbf{c}_i \neq \mathbf{0}$: $\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$

✍ : $\boldsymbol{R}_i$

$\mathbf{c}_i = \mathbf{0}$: $\boldsymbol{Q}'_i$

🍾 + 📄 + ✍ →

Verifier

Challenge

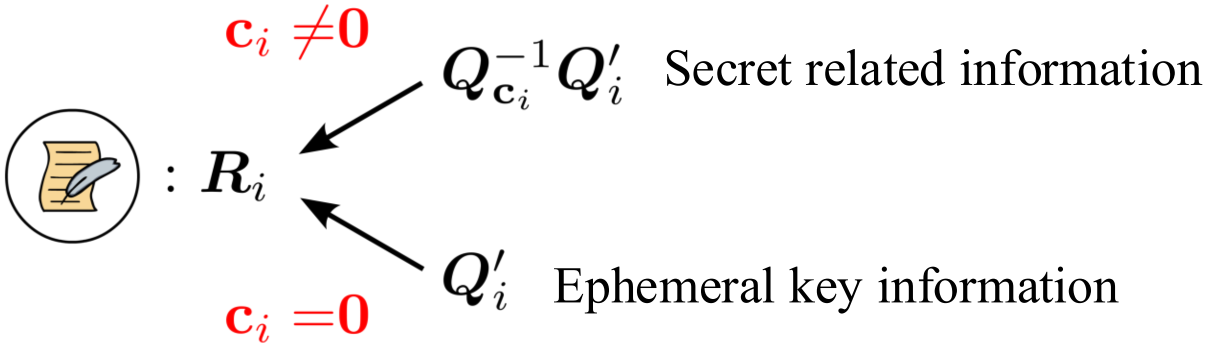
$\mathbf{Hash}($ 🍾, 📄 $) \longrightarrow$ 🧩

$\boldsymbol{G}'_i = S(\boldsymbol{G}_{\mathbf{c}_i}\boldsymbol{R}_i)$

Check :

🍾 $= \mathbf{Hash}(\boldsymbol{G}'_1, \boldsymbol{G}'_2, \cdots, \boldsymbol{G}'_t)$

Target of Our Fault Attack



$\mathbf{c}_i \neq \mathbf{0}$

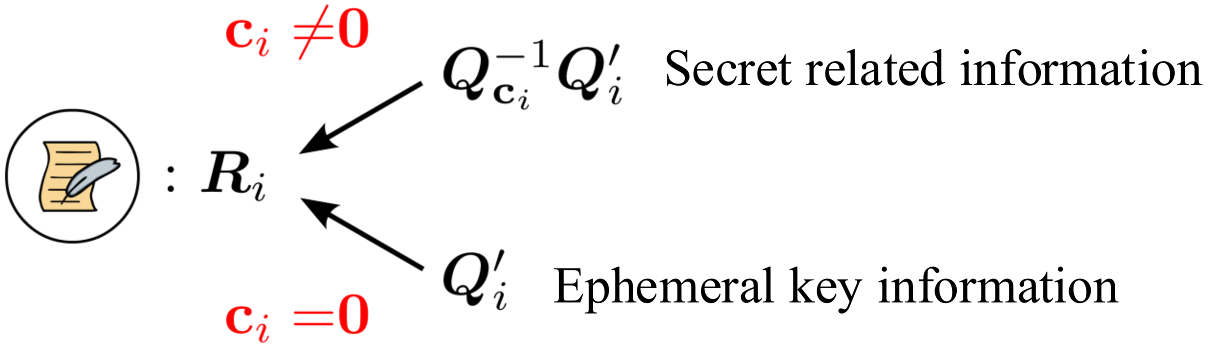
$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$ Secret related information

: $\boldsymbol{R}_i$

$\boldsymbol{Q}'_i$ Ephemeral key information

$\mathbf{c}_i = \mathbf{0}$

Target of Our Fault Attack



$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$ Secret related information

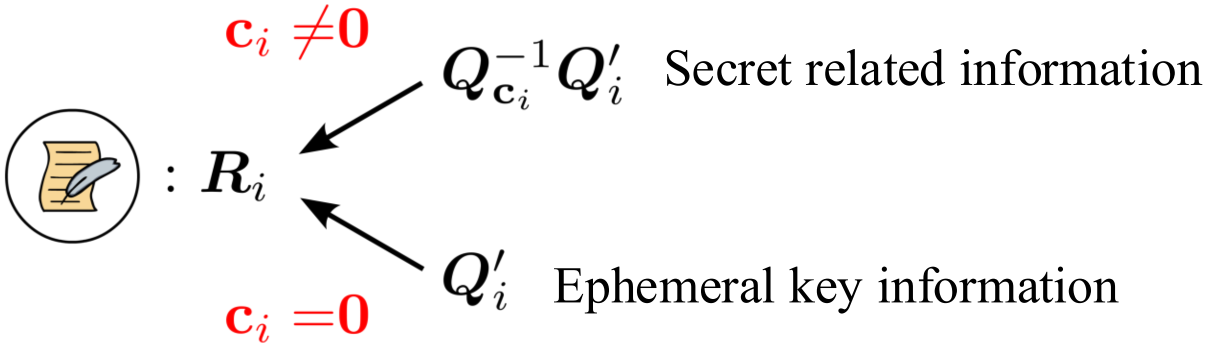
$: \boldsymbol{R}_i$

$\boldsymbol{Q}'_i$ Ephemeral key information

$\mathbf{c}_i = \mathbf{0}$

- For a fixed $\mathbf{c}_\mathbf{i}$, we will not get both $\mathbf{Q'}_\mathbf{i}$ or $\mathbf{Q}^{\mathbf{-1}}{}_{\mathbf{ci}}\mathbf{Q'}_\mathbf{i}$

Target of Our Fault Attack



$\mathbf{c}_i \neq \mathbf{0}$

$: \boldsymbol{R}_i$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$ Secret related information

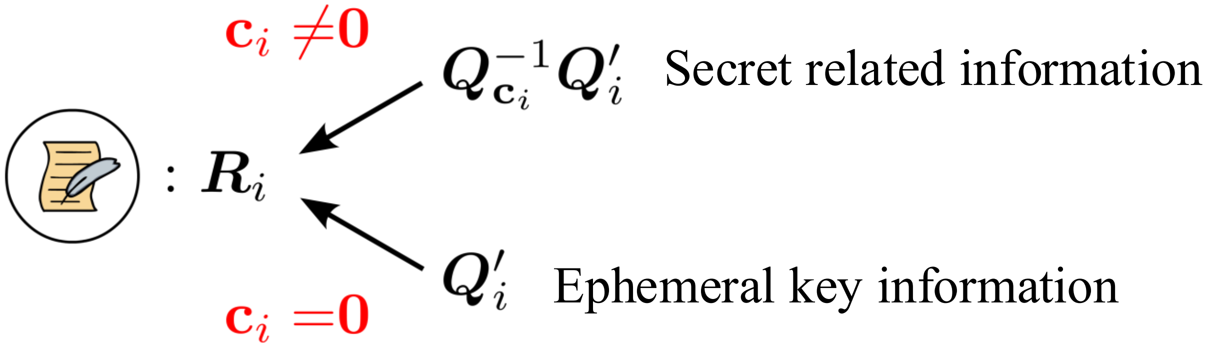
$\boldsymbol{Q}'_i$ Ephemeral key information

$\mathbf{c}_i = \mathbf{0}$

- For a fixed $\mathbf{c}_i$, we will not get both $\mathbf{Q'}_i$ or $\mathbf{Q}^{-1}{}_{\mathbf{ci}}\mathbf{Q'}_i$

- If we have both $\mathbf{Q'}_i$ and $\mathbf{Q}^{-1}{}_{\mathbf{ci}}\mathbf{Q'}_i$, then the secret $\mathbf{Q}_{\mathbf{ci}}\mathbf{=Q'}_i\mathbf{(Q}^{-1}{}_{\mathbf{ci}}\mathbf{Q'}_i\mathbf{)}^{-1}$ will be recovered.

Target of Our Fault Attack



$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$ Secret related information

$: \boldsymbol{R}_i$

$\boldsymbol{Q}'_i$ Ephemeral key information

$\mathbf{c}_i = \mathbf{0}$

- For a fixed $\mathbf{c}_i$, we will not get both $\mathbf{Q'}_i$ or $\mathbf{Q}^{-1}_{ci}\mathbf{Q'}_i$

- If we have both $\mathbf{Q'}_i$ and $\mathbf{Q}^{-1}_{ci}\mathbf{Q'}_i$, then the secret $\mathbf{Q}_{ci}=\mathbf{Q'}_i(\mathbf{Q}^{-1}_{ci}\mathbf{Q'}_i)^{-1}$ will be recovered.

Target

Get a pair $(\mathbf{Q'}_i,\mathbf{Q}^{-1}_{ci}\mathbf{Q'}_i)$

Fault Assumption and Result

- Our fault assumption is we can change the value of a location from 1 to 0

Fault Assumption and Result

- Our fault assumption is we can change the value of a location from 1 to 0

- This can be achieved by any one of
  - bit flip fault
  - stuck at zero fault
  - instruction skip fault

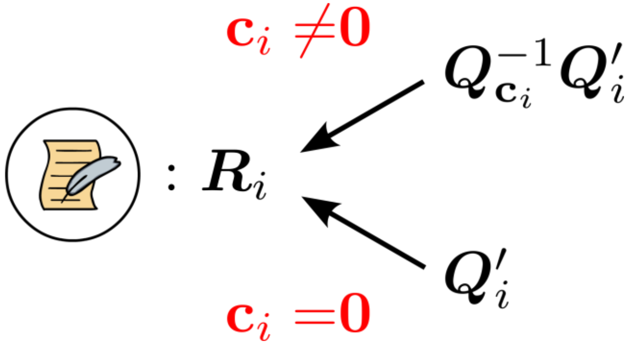
Fault Assumption and Result

- Our fault assumption is we can change the value of a location from 1 to 0

- This can be achieved by any one of
  - bit flip fault
  - stuck at zero fault
  - instruction skip fault

Scheme name	Security level	Parameter set	Required faults for complete secret recovery
LESS	1	Less-1b	**1**
		Less-1i	**1**
		Less-1s	**2**
	3	Less-3b	**1**
		Less-3s	**1**
	5	Less-5b	**1**
		Less-5s	**1**
CROSS	1, 3, 5	CROSS-R-SDP	**1**
		CROSS-R-SDPG	**1**

Overview of LESS



$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$

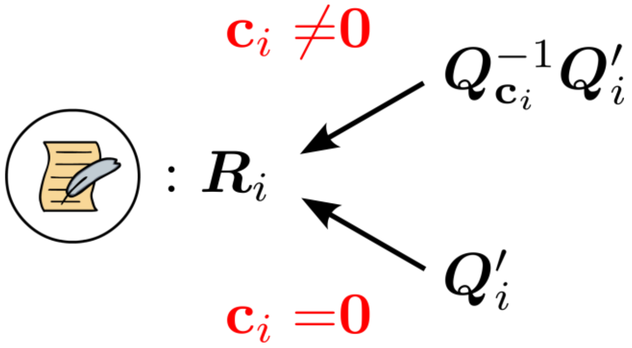
$: \boldsymbol{R}_i$

$\boldsymbol{Q}'_i$

$\mathbf{c}_i = \mathbf{0}$

Overview of LESS

- Each **Q'**$_i$ is generated from a random seed **seed**$_i$



$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$

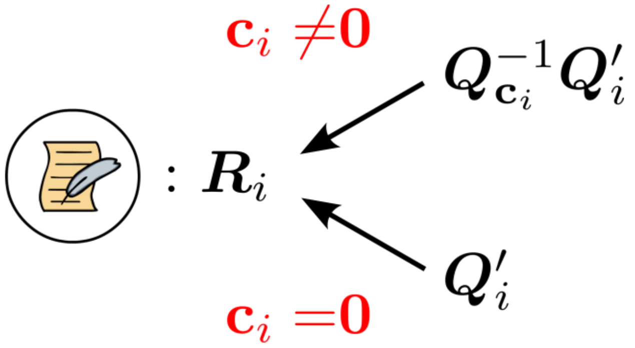
: $\boldsymbol{R}_i$

$\boldsymbol{Q}'_i$

$\mathbf{c}_i = \mathbf{0}$

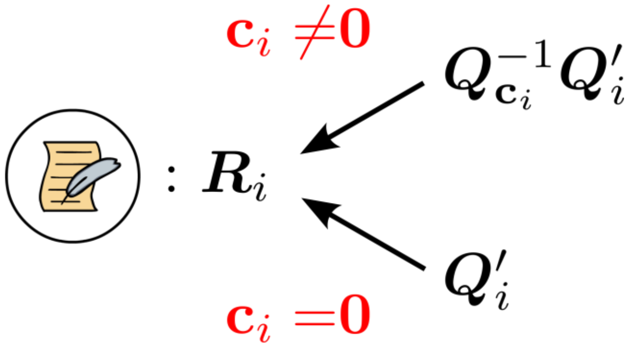
Overview of LESS

- Each **Q'$_i$** is generated from a random seed **seed$_i$**
- The challenge **c=c$_1$,c$_2$,..,c$_t$** is of fixed weight w


$\mathbf{c}_i \neq \mathbf{0}$
$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$
$: \boldsymbol{R}_i$
$\boldsymbol{Q}'_i$
$\mathbf{c}_i = \mathbf{0}$

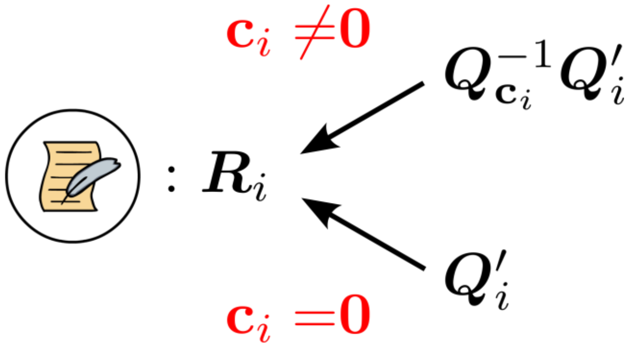
Overview of LESS

- Each $\mathbf{Q'}_\mathbf{i}$ is generated from a random seed $\mathbf{seed}_\mathbf{i}$
- The challenge $\mathbf{c}=\mathbf{c}_1,\mathbf{c}_2,..,\mathbf{c}_t$ is of fixed weight w
- The response must contain information of (t-w) many seeds of $\mathbf{Q'}_\mathbf{i}$


$\mathbf{c}_i \neq \mathbf{0}$
$Q_{\mathbf{c}_i}^{-1} Q'_i$
$: \boldsymbol{R}_i$
Q'_i
$\mathbf{c}_i = \mathbf{0}$

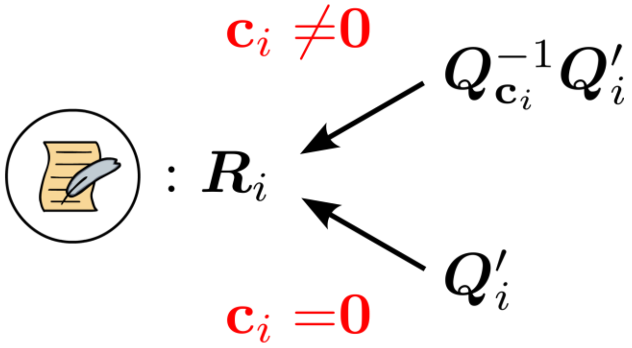
Overview of LESS

- Each **Q'**$_{\mathbf{i}}$ is generated from a random seed **seed**$_{\mathbf{i}}$
- The challenge **c=c**$_{\mathbf{1}}$**,c**$_{\mathbf{2}}$**,..,c**$_{\mathbf{t}}$ is of fixed weight w
- The response must contain information of (t-w) many seeds of **Q'**$_{\mathbf{i}}$
- For parameter, LESS-5b signature size 74960 Bytes



Overview of LESS

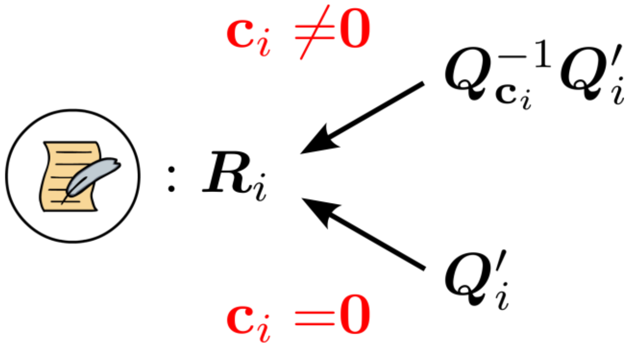
- Each $\mathbf{Q'}_\mathrm{i}$ is generated from a random seed $\mathbf{seed}_\mathrm{i}$
- The challenge $\mathbf{c=c_1,c_2,..,c_t}$ is of fixed weight w
- The response must contain information of (t-w) many seeds of $\mathbf{Q'}_\mathrm{i}$
- For parameter, LESS-5b signature size 74960 Bytes
- LESS uses Goldreich-Goldwasser-Micali (GGM) tree construction



$\mathbf{c}_i \neq \mathbf{0}$
$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$
$: \boldsymbol{R}_i$
$\boldsymbol{Q}'_i$
$\mathbf{c}_i = \mathbf{0}$

Overview of LESS

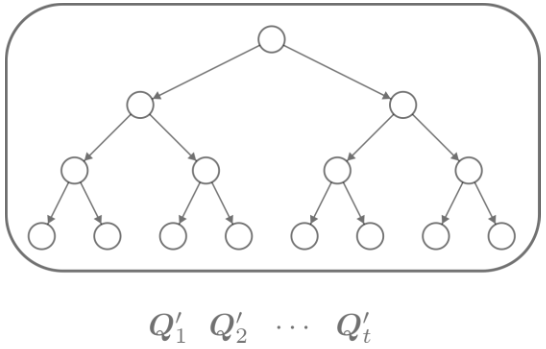
- Each $\mathbf{Q'}_{\mathbf{i}}$ is generated from a random seed $\mathbf{seed}_{\mathbf{i}}$
- The challenge $\mathbf{c}=\mathbf{c}_1,\mathbf{c}_2,..,\mathbf{c}_t$ is of fixed weight w
- The response must contain information of (t-w) many seeds of $\mathbf{Q'}_{\mathbf{i}}$
- For parameter, LESS-5b signature size 74960 Bytes
- LESS uses Goldreich-Goldwasser-Micali (GGM) tree construction
- Compressed signatures size: ~32500 Bytes (~57% reduced)


$\mathbf{c}_i \neq \mathbf{0}$
$Q_{\mathbf{c}_i}^{-1}Q'_i$
$: \boldsymbol{R}_i$
Q'_i
$\mathbf{c}_i = \mathbf{0}$

Summary of Signature Compression

Step 1

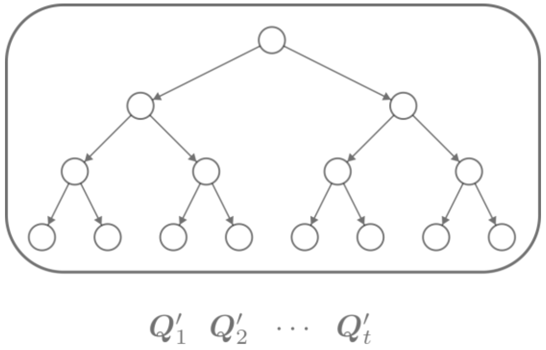
Generate a GGM tree (seedtree) and generate all the ephimeral keys Q_1', Q_2', … ,Q_t'



Summary of Signature Compression

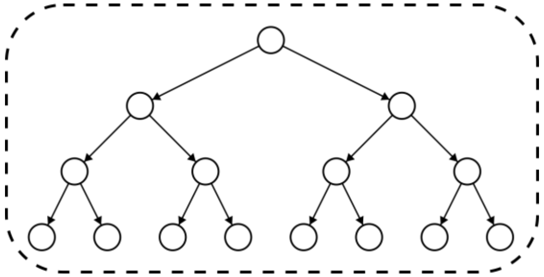
Step 1

Generate a GGM tree (seedtree) and generate all the ephimeral keys Q_1', Q_2', … ,Q_t'



Step 2

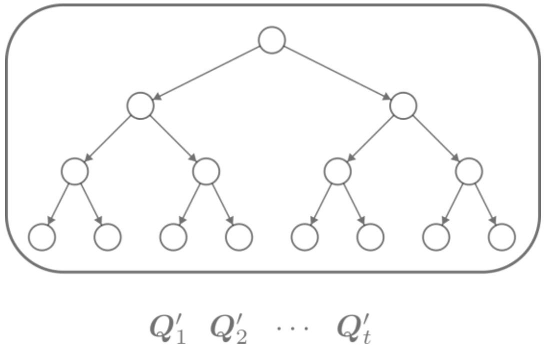
Generate a reference tree based on the challenge



Summary of Signature Compression

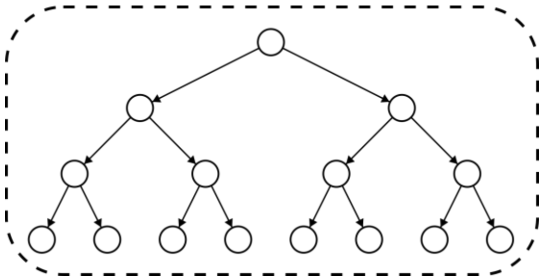
Step 1

Generate a GGM tree (seedtree) and generate all the ephimeral keys Q_1', Q_2', … ,Q_t'



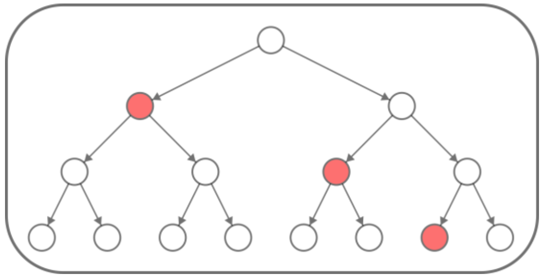
Step 2

Generate a reference tree based on the challenge



Step 3

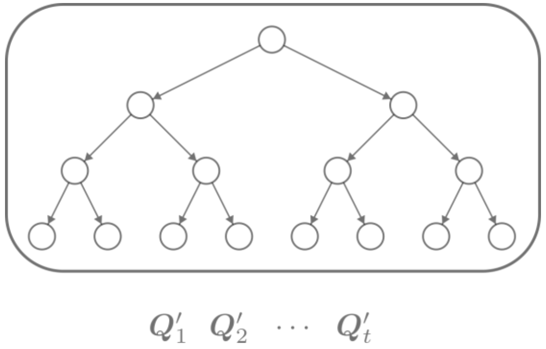
Decide the nodes of the seedtree to be published based on reference tree



Summary of Signature Compression

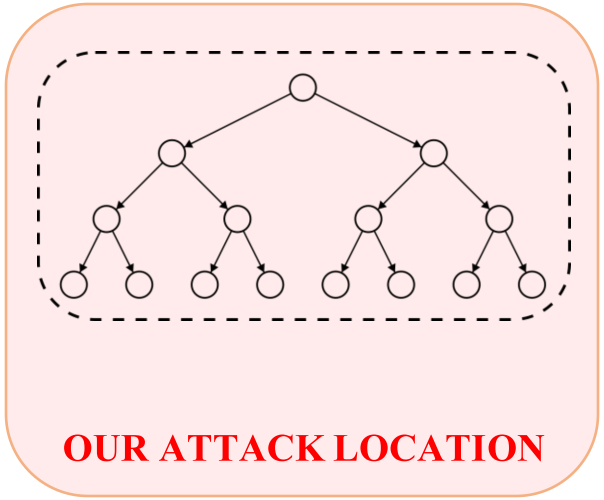
Step 1

Generate a GGM tree (seedtree) and generate all the ephimeral keys Q_1', Q_2', … ,Q_t'



Step 2

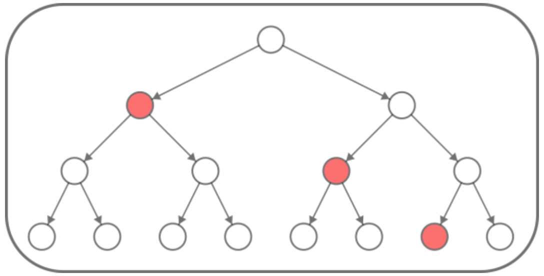
Generate a reference tree based on the challenge



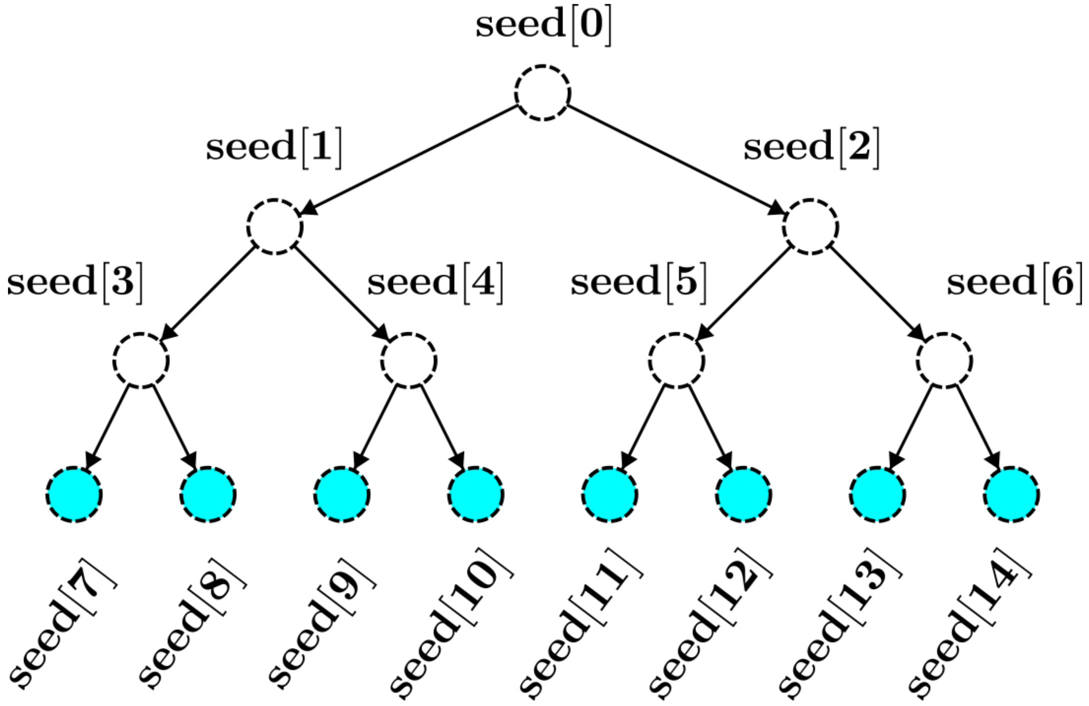
OUR ATTACK LOCATION

Step 3

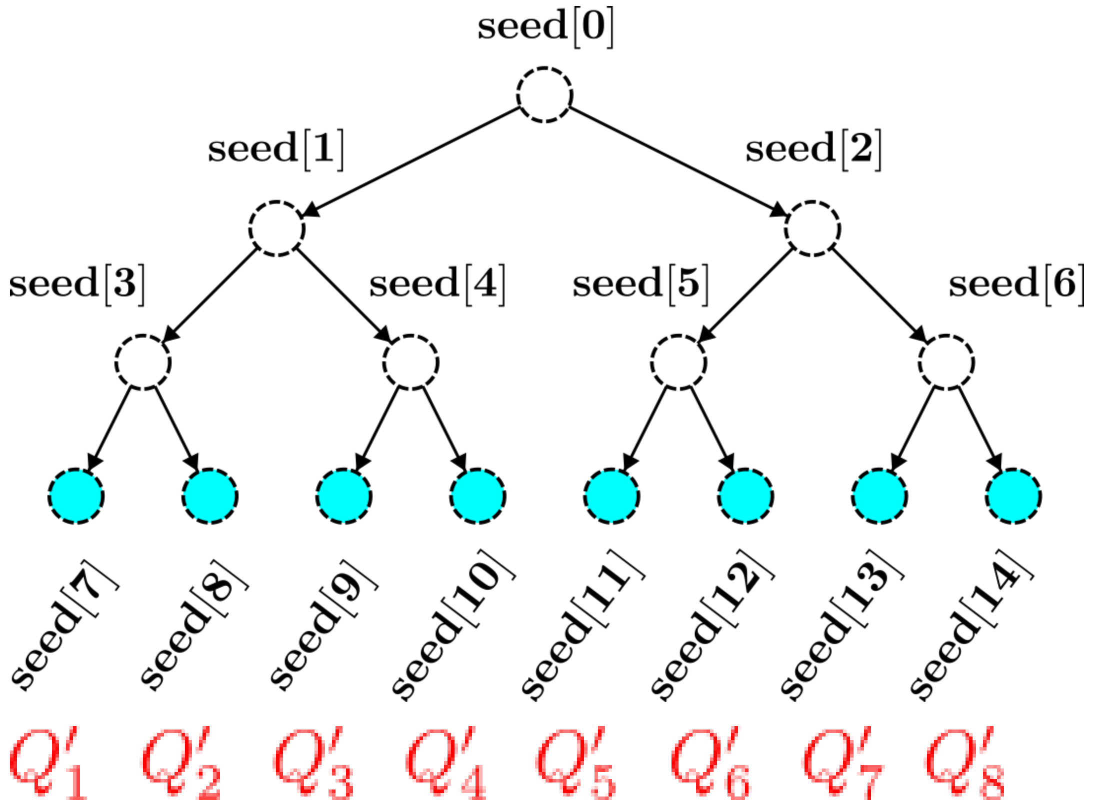
Decide the nodes of the seedtree to be published based on reference tree



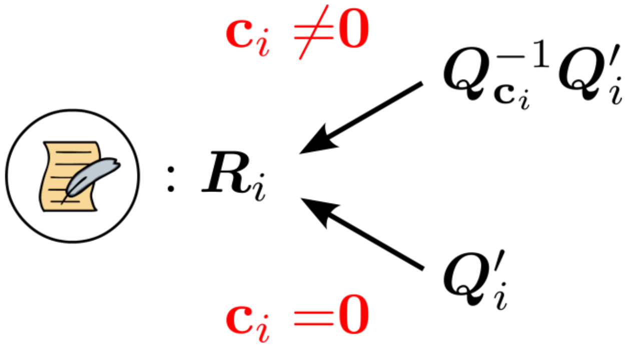
Seed Tree Generation



Seed Tree Generation



Compression Technique



$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$

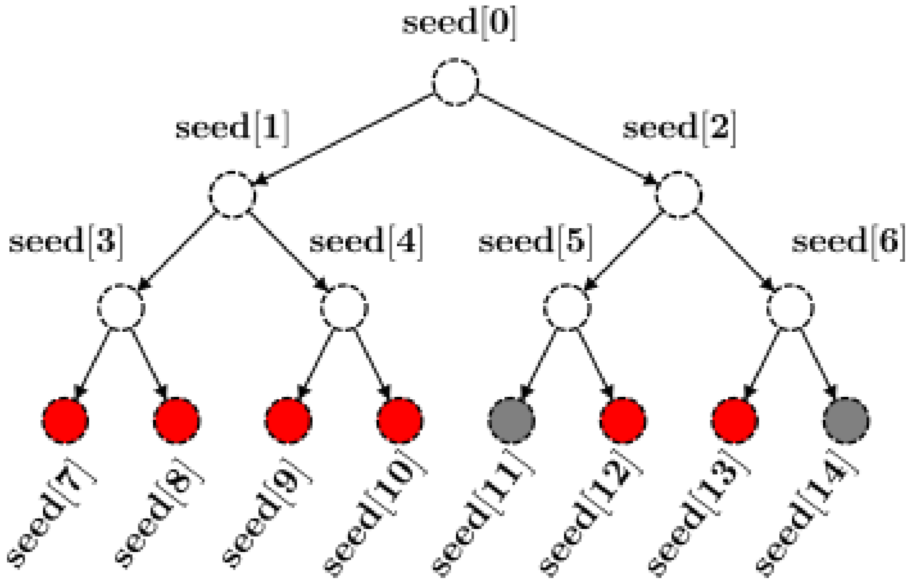
$: \boldsymbol{R}_i$

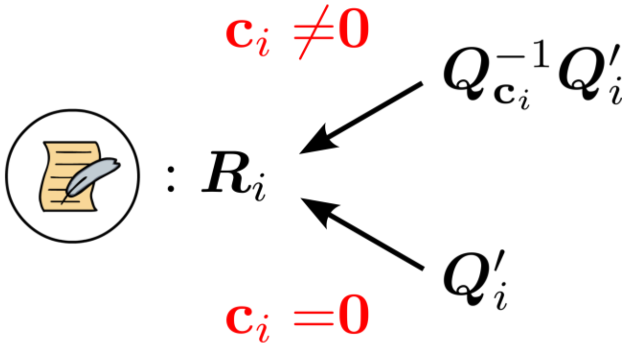
$\boldsymbol{Q}'_i$

$\mathbf{c}_i = \mathbf{0}$

Suppose **c=(0,0,0,0,1,0,0,2)**

Compression Technique





$\mathbf{c}_i \neq \mathbf{0}$

$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}_i'$

: $\boldsymbol{R}_i$

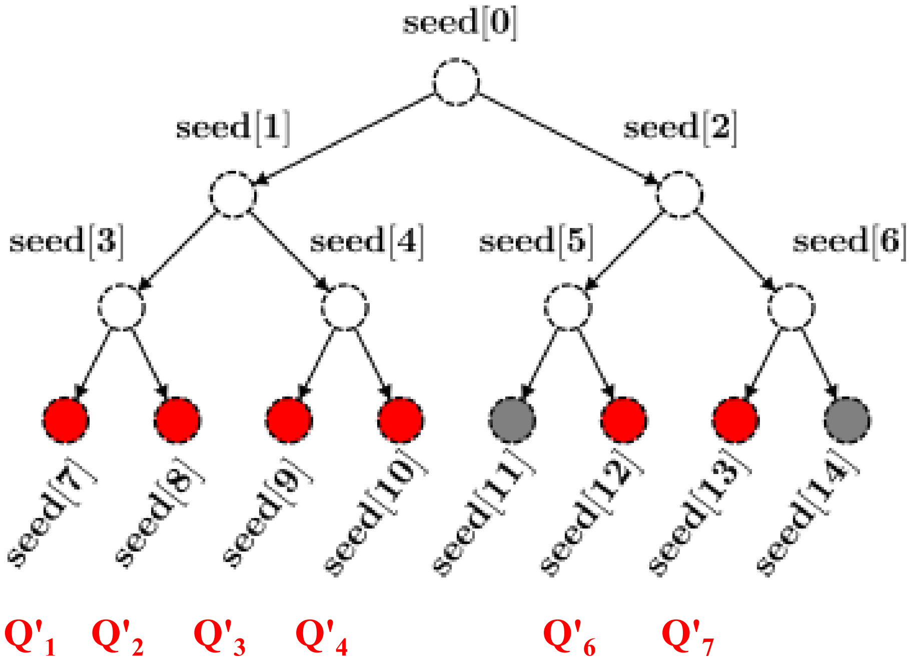
$\boldsymbol{Q}_i'$

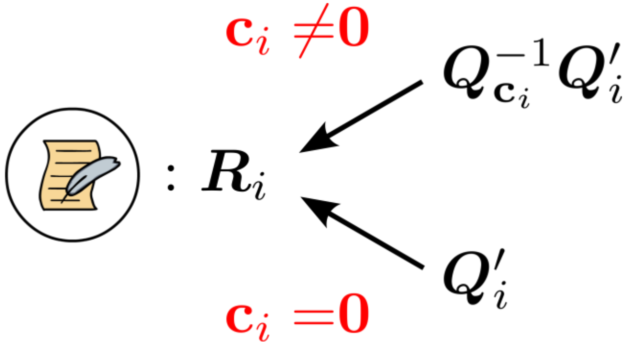
$\mathbf{c}_i = \mathbf{0}$

Suppose **c=(0,0,0,0,1,0,0,2)**

● : to publish $\mathbf{Q'_i}$, ● : to hide $\mathbf{Q'_i}$

Compression Technique

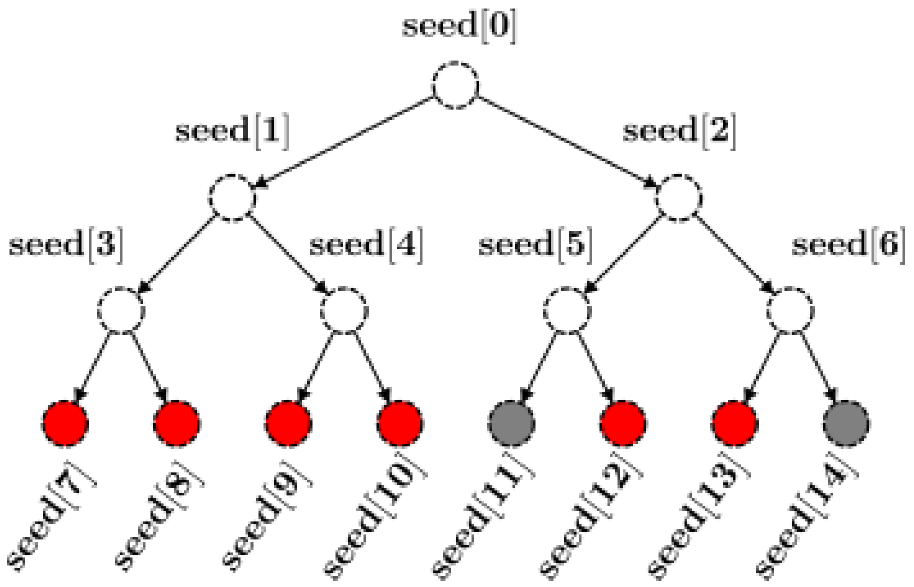



$\mathbf{c}_i \neq \mathbf{0}$
$\boldsymbol{Q}_{\mathbf{c}_i}^{-1}\boldsymbol{Q}'_i$
$: \boldsymbol{R}_i$
$\boldsymbol{Q}'_i$
$\mathbf{c}_i = \mathbf{0}$

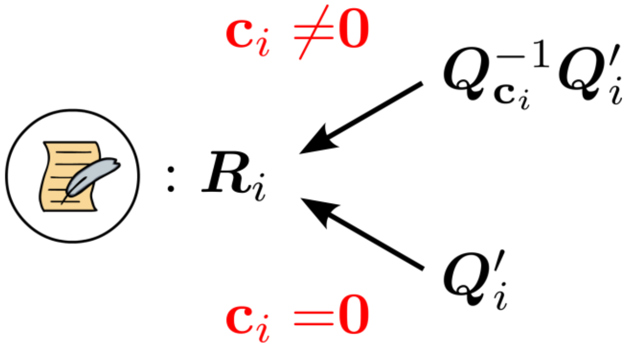
Suppose **c=(0,0,0,0,1,0,0,2)**

🔴 : to publish $\mathbf{Q'_i}$, ⚫ : to hide $\mathbf{Q'_i}$

Compression Technique



Q'_1 Q'_2 Q'_3 Q'_4 $Q^{-1}_1Q'_5$ Q'_6 Q'_7 $Q^{-1}_2Q'_8$



$\mathbf{c}_i \neq \mathbf{0}$ $Q_{\mathbf{c}_i}^{-1}Q'_i$

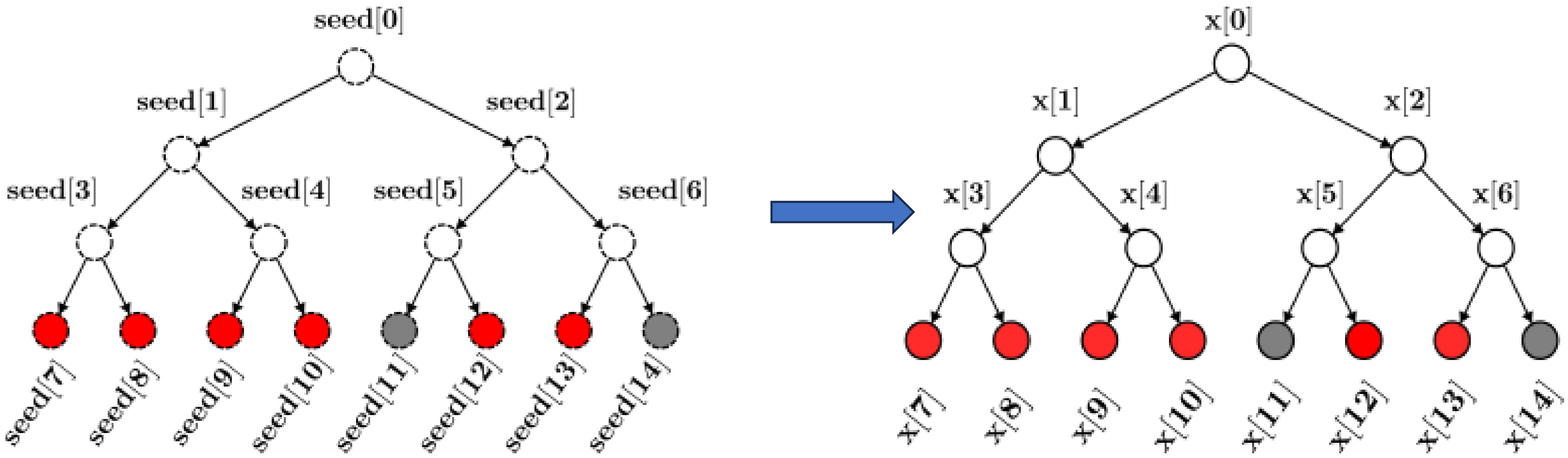
$: R_i$

$\mathbf{c}_i = \mathbf{0}$ Q'_i

Suppose **c=(0,0,0,0,1,0,0,2)**

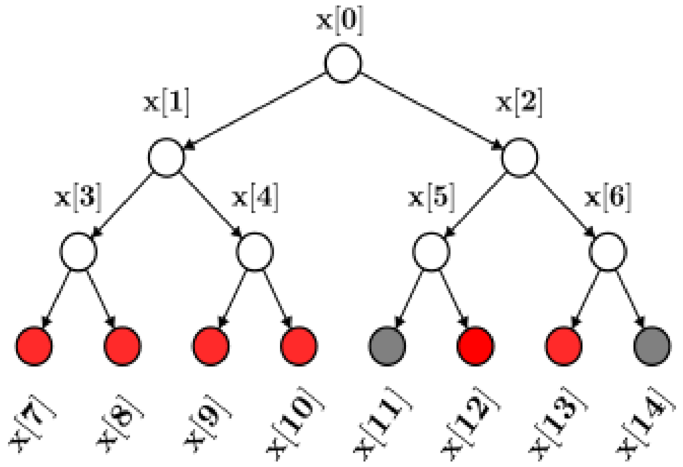
● : to publish $\mathbf{Q'_i}$, ● : to hide $\mathbf{Q'_i}$

Compression Technique



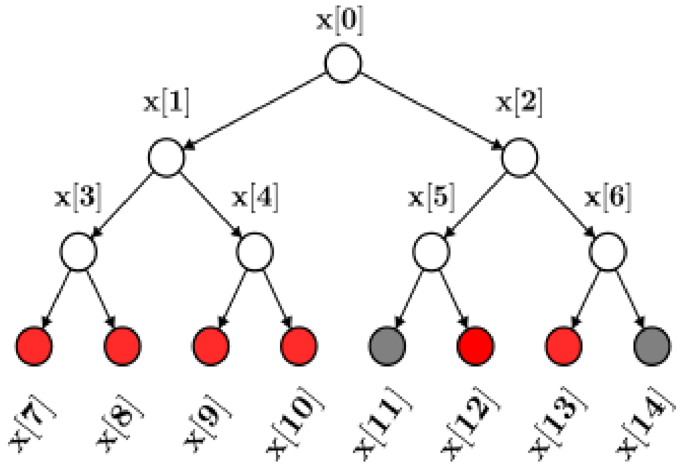
🔴: to publish **Q'**$_i$, ⚫ : to hide **Q'**$_i$

Compression Technique



🔴: to publish **Q'**$_{\mathbf{i}}$, ● : to hide **Q'**$_{\mathbf{i}}$

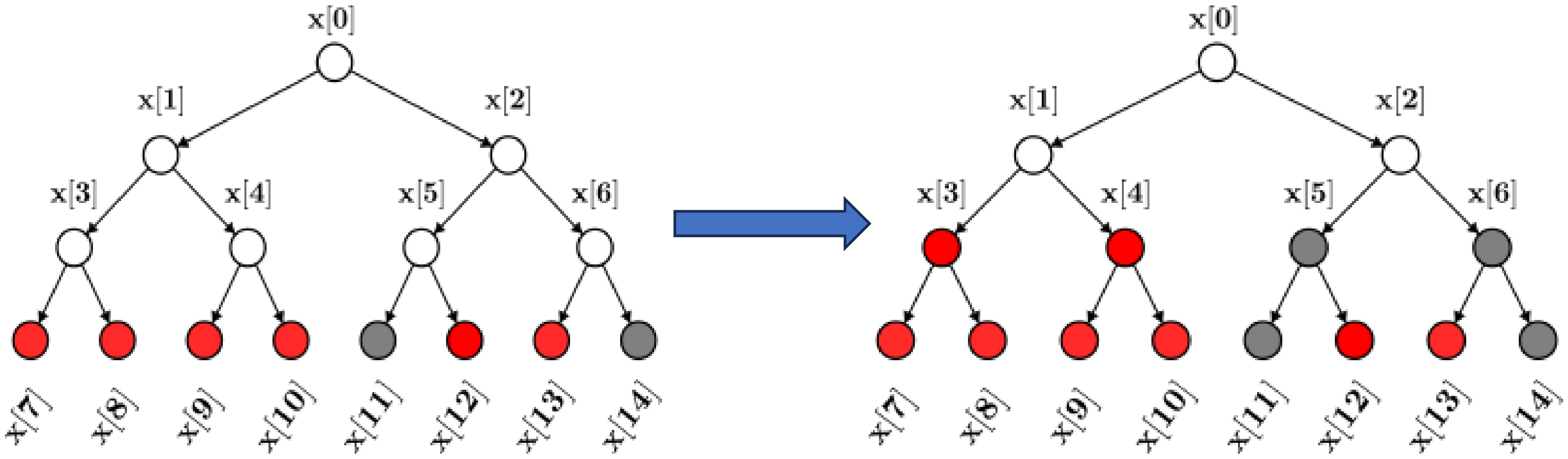
Compression Technique



🔴: to publish $\mathbf{Q'}_i$, ● : to hide $\mathbf{Q'}_i$

i-th node: 🔴 if both children are 🔴

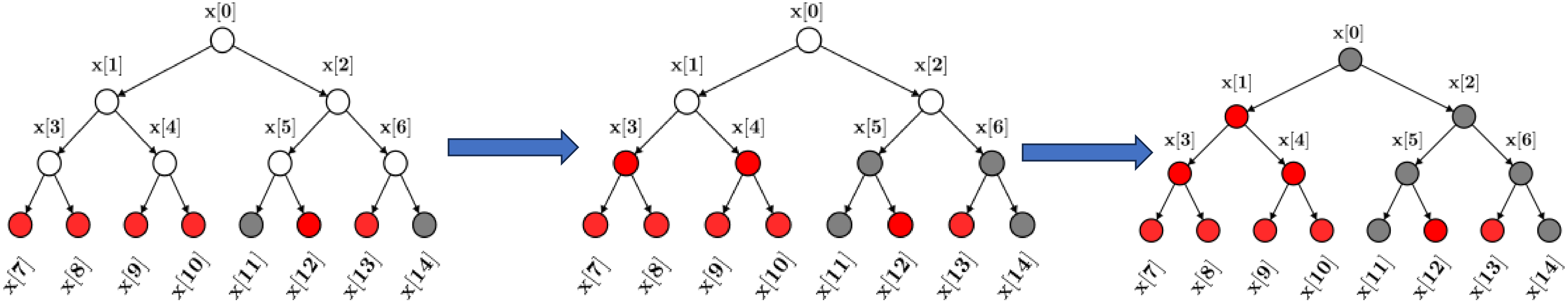
Compression Technique



🔴: to publish **Q'**$_i$, ⚫ : to hide **Q'**$_i$

i-th node: 🔴 if both children are 🔴

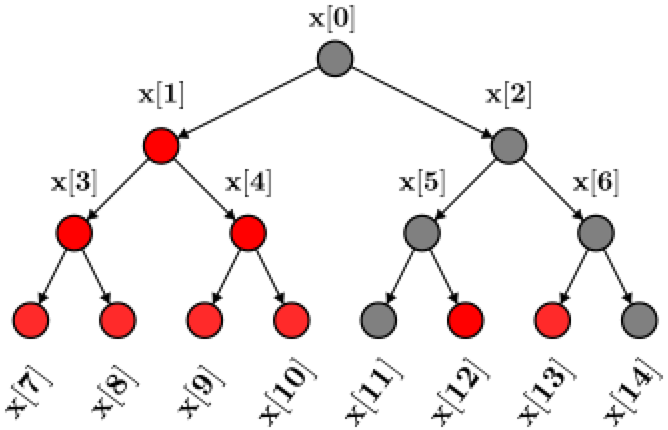
Compression Technique



🔴: to publish **Q'**$_i$, ⚫ : to hide **Q'**$_i$

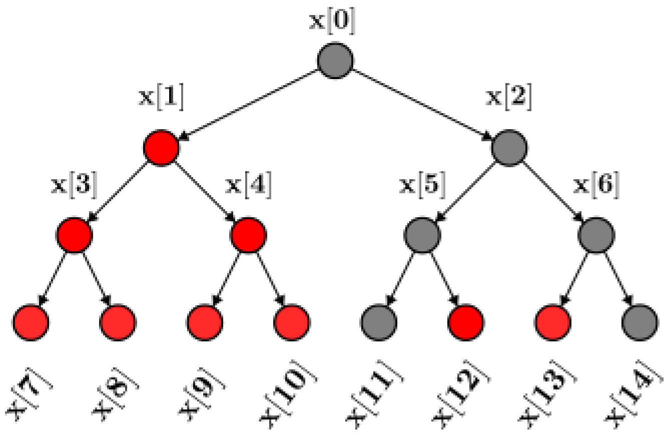
i-th node: 🔴 if both children are 🔴

Compression Technique



🔴 : to publish **Q'**$_i$, ● : to hide **Q'**$_i$

Compression Technique

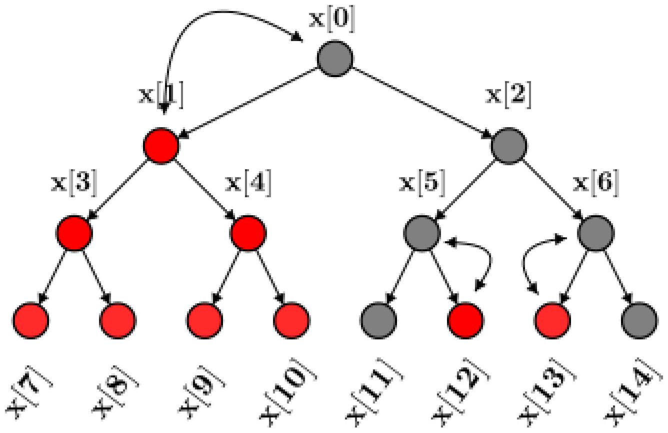


🔴 : to publish $\mathbf{Q'}_i$, ⚫ : to hide $\mathbf{Q'}_i$

Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

Compression Technique

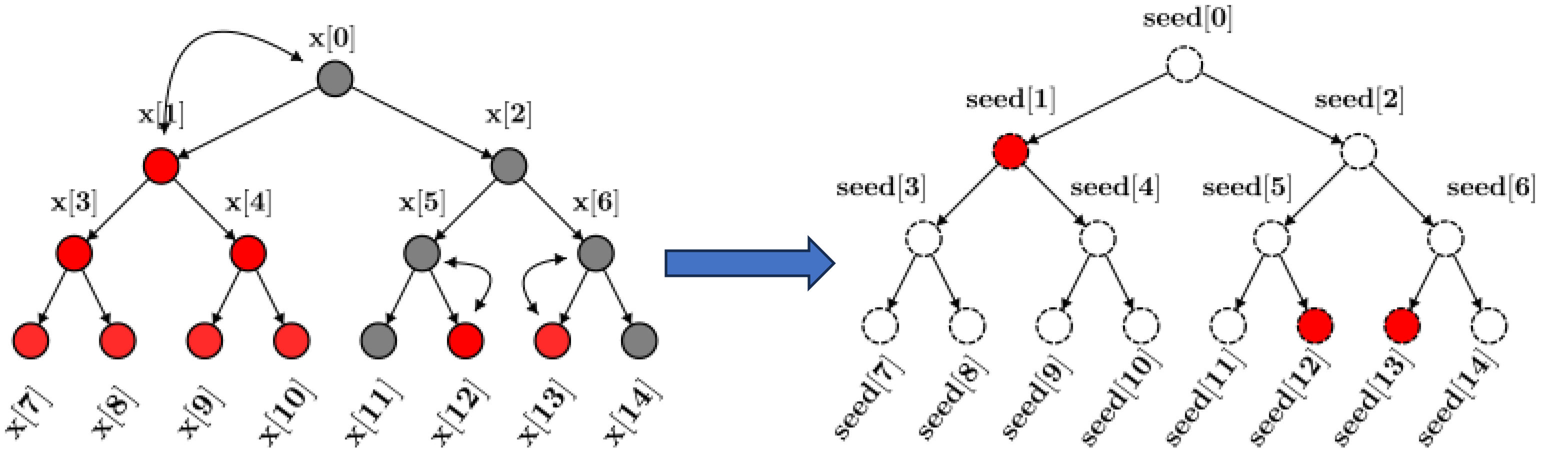


🔴 : to publish **Q'**$_i$, ⚫ : to hide **Q'**$_i$

Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

Compression Technique

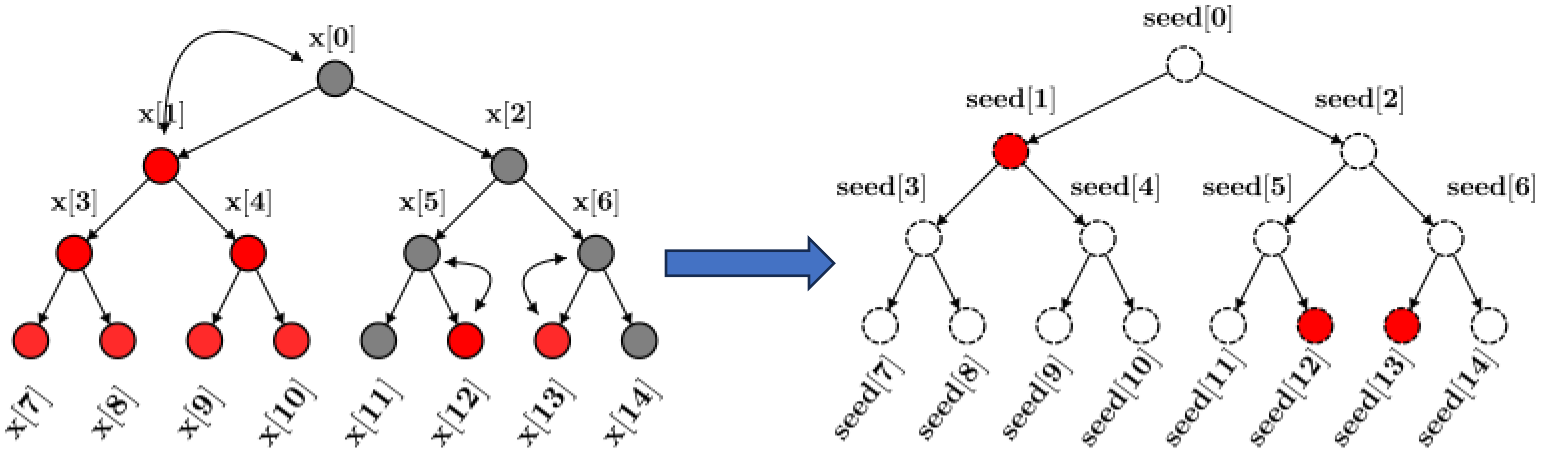


🔴 : to publish **Q'**$_i$, ● : to hide **Q'**$_i$

Check: i-th node: 🔴 and its parent: ● ?

If yes: publish seed[i]

Compression Technique

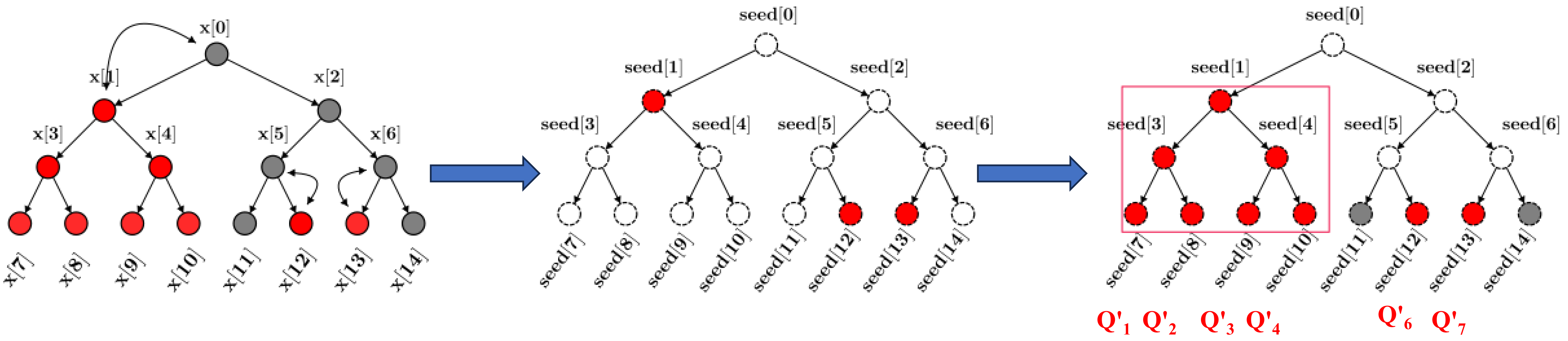


🔴 : to publish $\mathbf{Q'}_i$, ⚫ : to hide $\mathbf{Q'}_i$ publish seed[1],seed[12],seed[13]

Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

Compression Technique

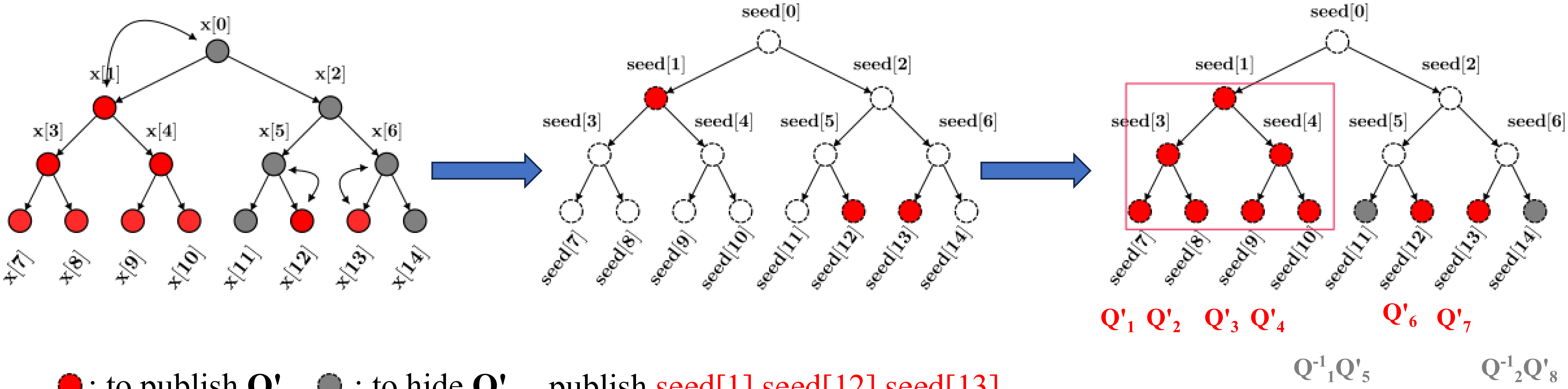


🔴: to publish **Q'**$_i$, ⚫ : to hide **Q'**$_i$ publish seed[1],seed[12],seed[13]

Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

Compression Technique

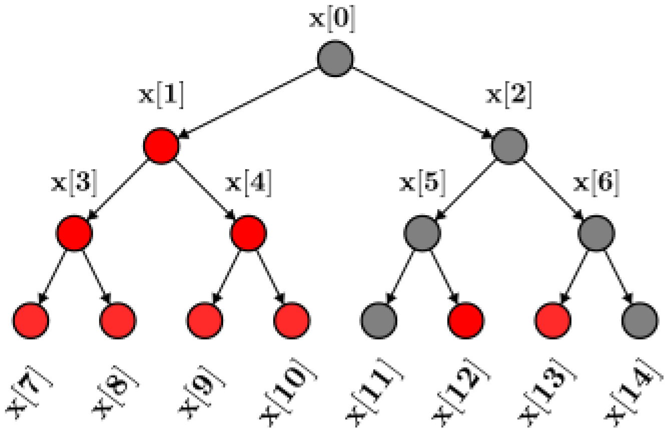


🔴 : to publish $\mathbf{Q'_i}$, ⚫ : to hide $\mathbf{Q'_i}$ publish seed[1],seed[12],seed[13]

Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

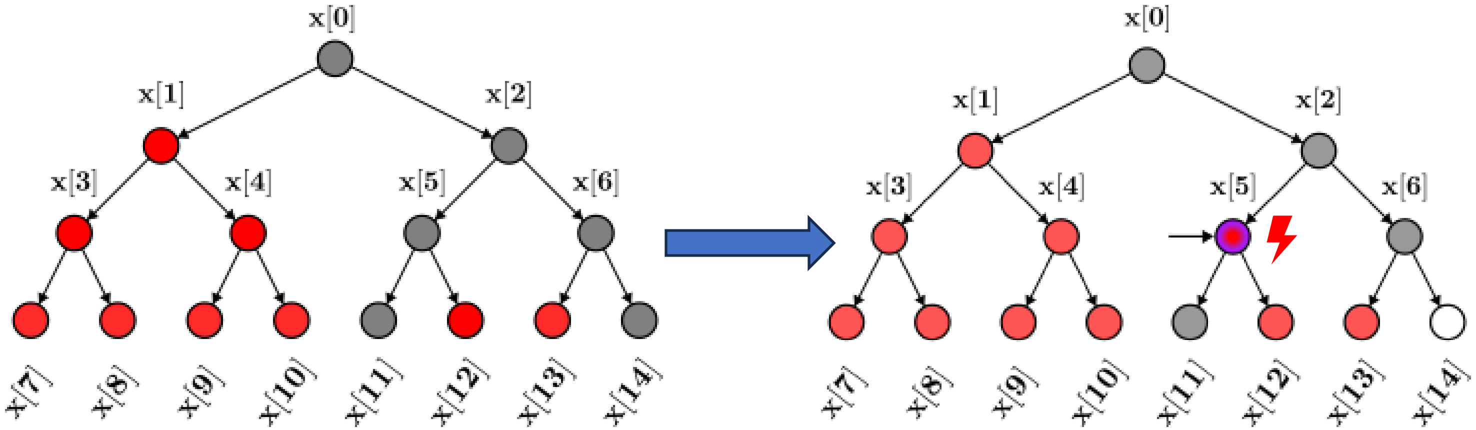
Our Attack Location



Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

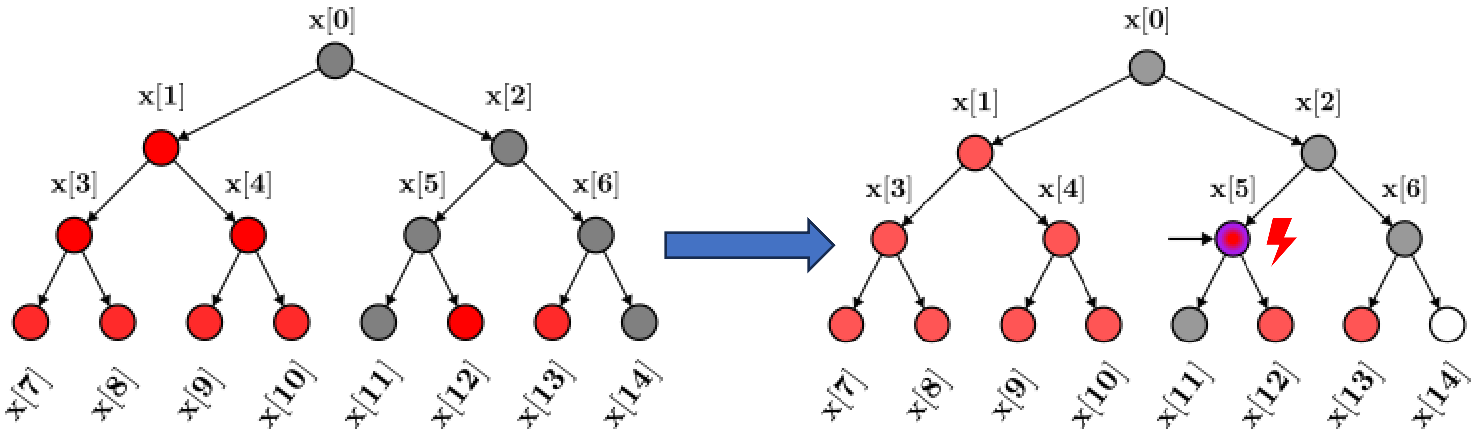
Our Attack Location



Check: i-th node: 🔴 and its parent: ⚫ ?

If yes: publish seed[i]

Our Attack Location

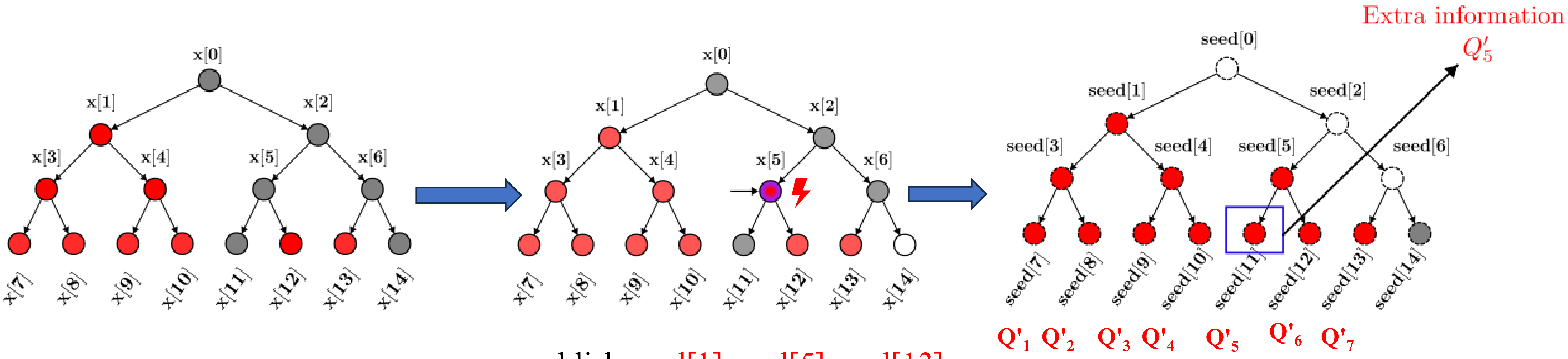


publish seed[1],seed[5],seed[13]

Check: i-th node: ● and its parent: ● ?

If yes: publish seed[i]

Our Attack Location

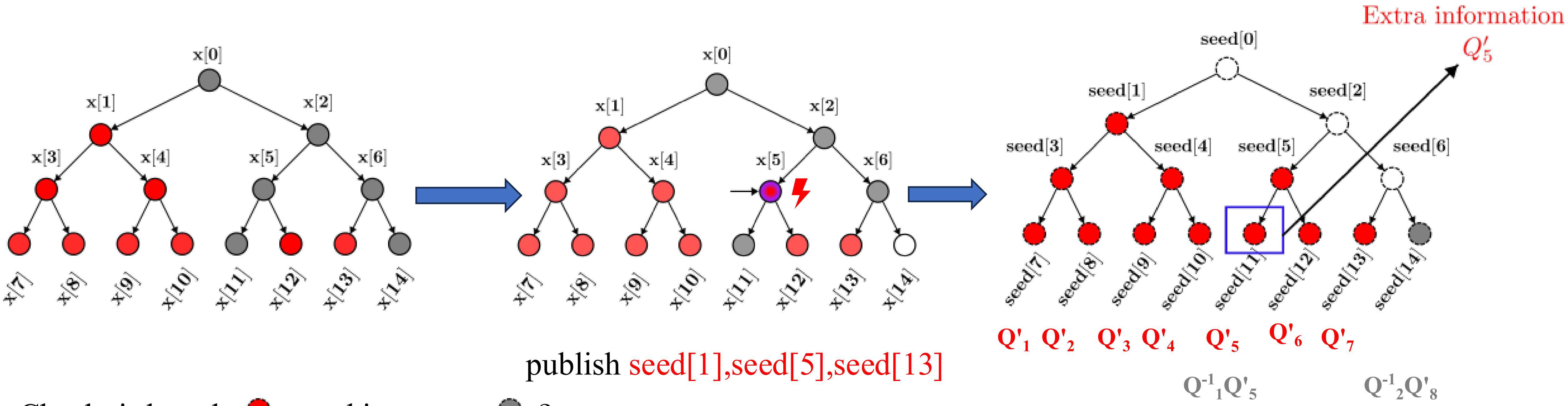


publish seed[1],seed[5],seed[13]

Check: i-th node: ● and its parent: ● ?

If yes: publish seed[i]

Our Attack Location

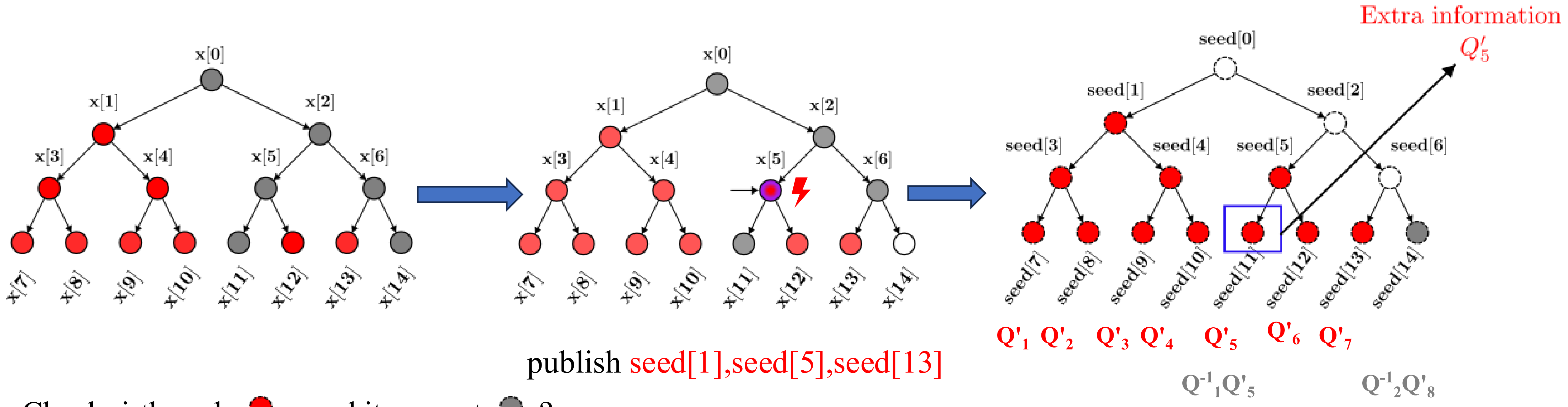


publish seed[1],seed[5],seed[13]

Check: i-th node: ● and its parent: ● ?

If yes: publish seed[i]

Our Attack Location



publish seed[1],seed[5],seed[13]

Check: i-th node: ● and its parent: ● ?

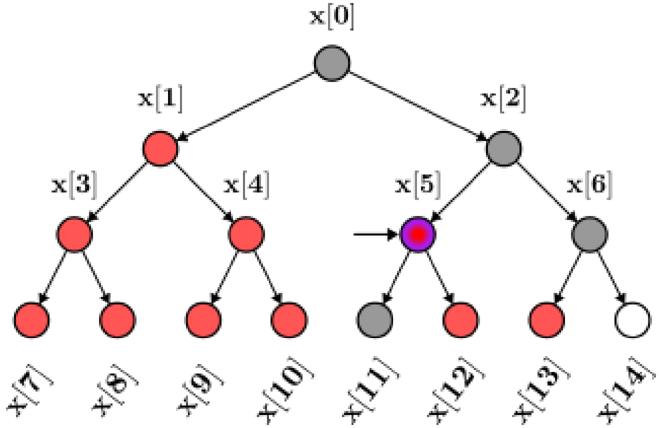
If yes: publish seed[i]

The secret Q_1 can be computed by

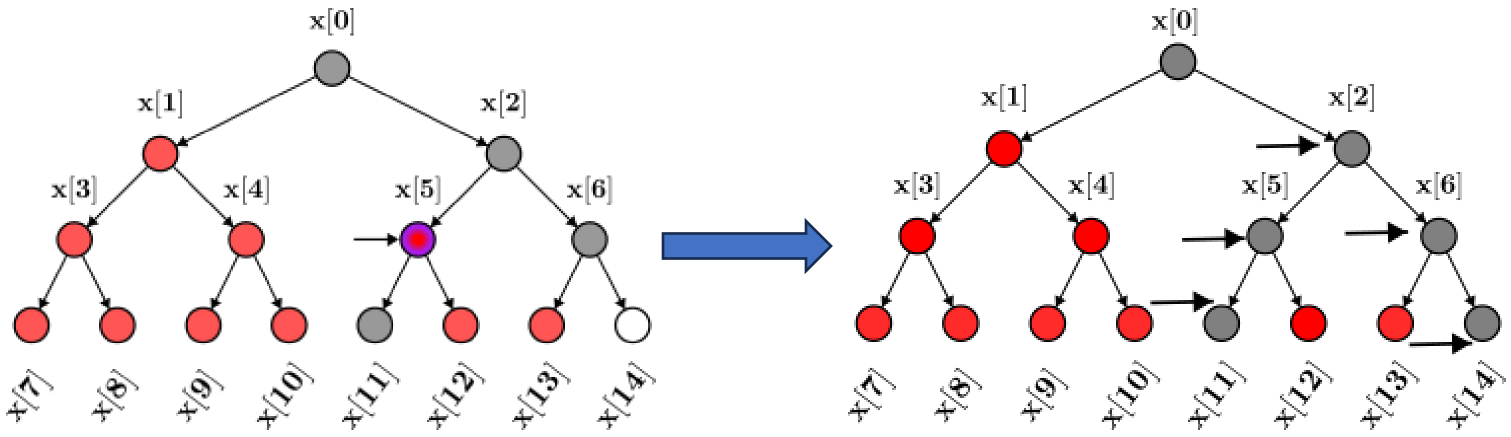
$$Q_1 = Q'_5 (Q^{-1}_1 Q'_5)^{-1}$$

Leaks secret information!!!!

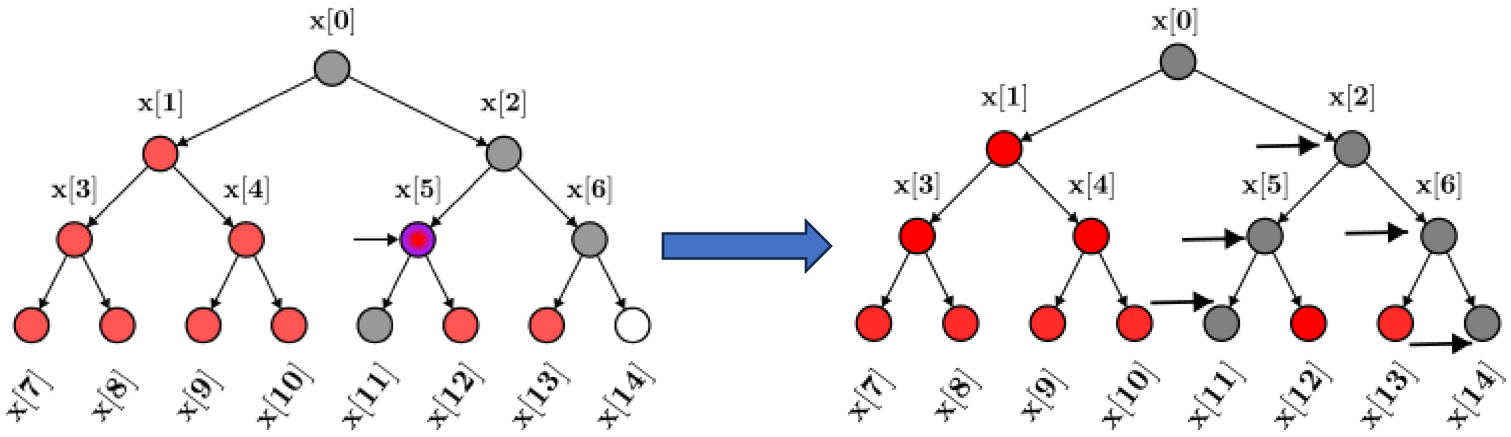
Generic Fault Location



Generic Fault Location

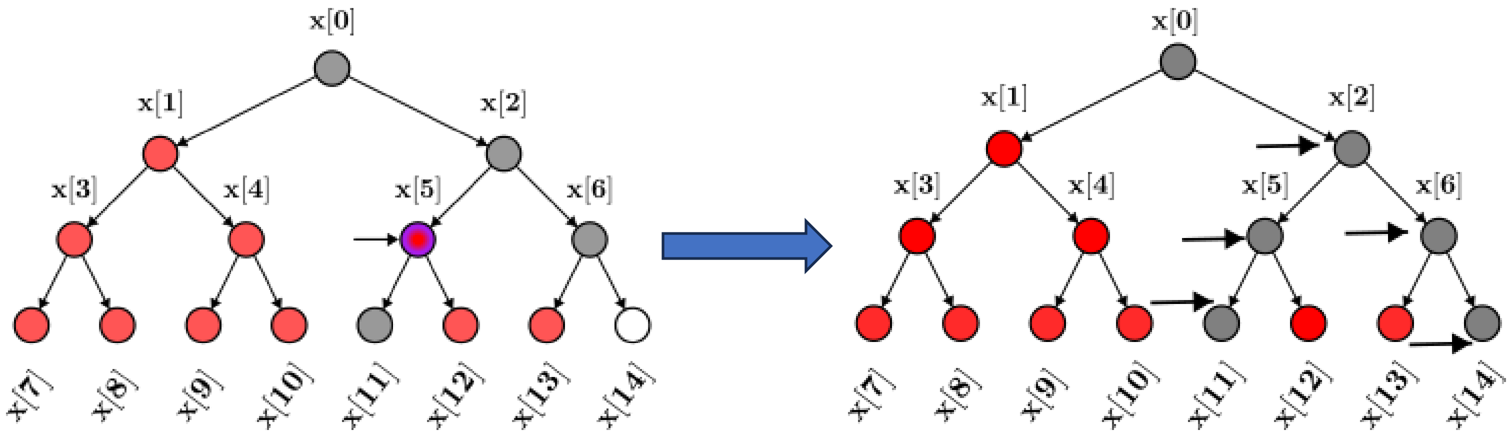


Generic Fault Location



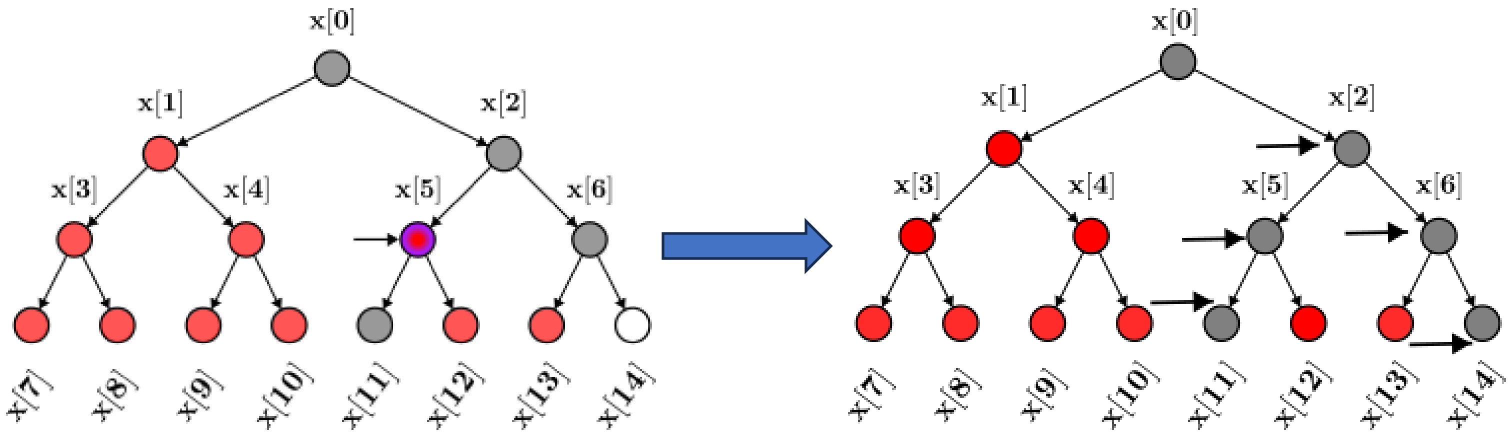
- Change of any grey node by injecting fault will leak the information of the secret

Generic Fault Location



- Change of any grey node by injecting fault will leak the information of the secret
- The nodes with higher heights would always give more informations about secret

Generic Fault Location



- Change of any grey node by injecting fault will leak the information of the secret
- The nodes with higher heights would always give more informations about secret
- For our result, we have targeted the location x[1]/x[2]

Fault Detection

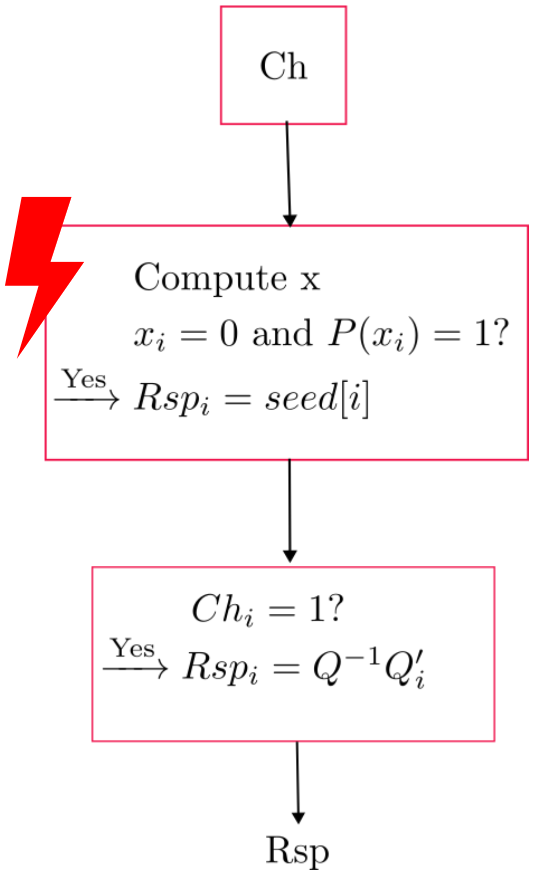
- After Fault injection we need to answer the questions:
  - Is the fault injected exactly at i-th location?
  - Is the fault injection successful?

Fault Detection

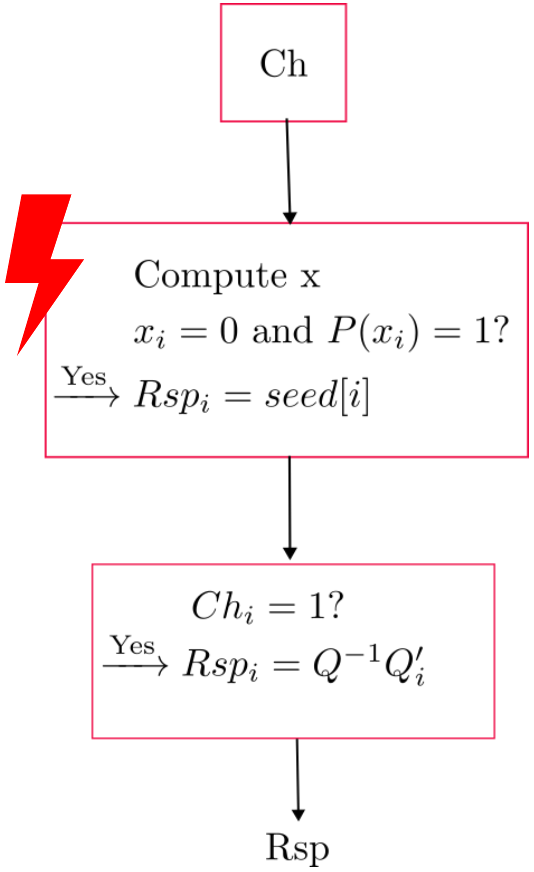
- After Fault injection we need to answer the questions:
  - Is the fault injected exactly at i-th location?
  - Is the fault injection successful?

- In our work, the fault detection method answers both of the above questions.

How to Prevent the Fault Attack?


Ch
Compute x
$x_i = 0$ and $P(x_i) = 1$?
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$
$Ch_i = 1$?
$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$
Rsp

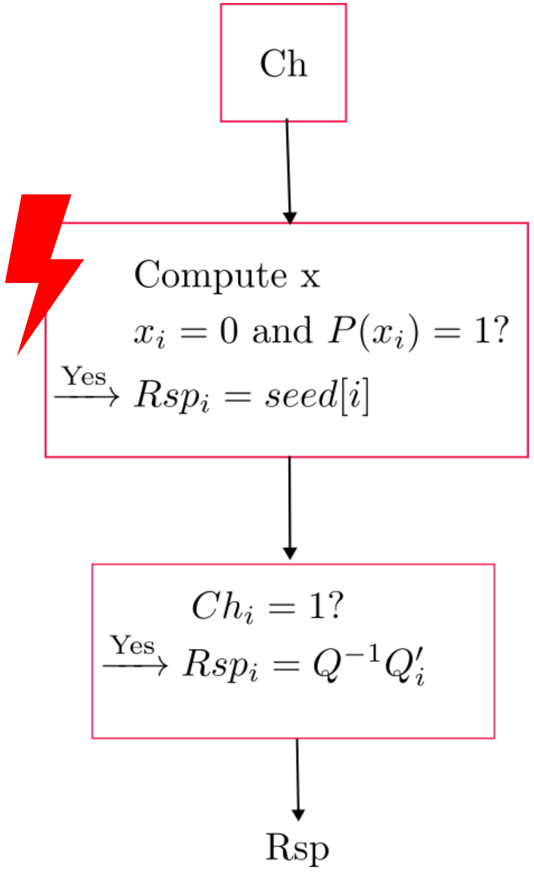
How to Prevent the Fault Attack?



Ch
Compute x
$x_i = 0$ and $P(x_i) = 1?$
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$
$Ch_i = 1?$
$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$
Rsp

- By injecting fault in the first part to get the value seed[i'] corresponding to Ch$_i$=1, which will not hamper the second part.

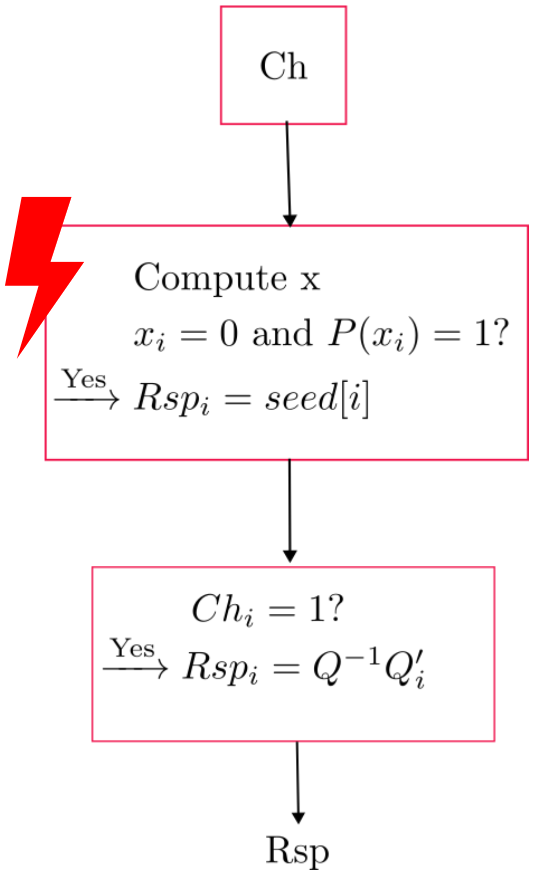
How to Prevent the Fault Attack?


Ch
Compute x
$x_i = 0$ and $P(x_i) = 1$?
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$
$Ch_i = 1$?
$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$
Rsp

- By injecting fault in the first part to get the value seed[i'] corresponding to Ch_i=1, which will not hamper the second part.

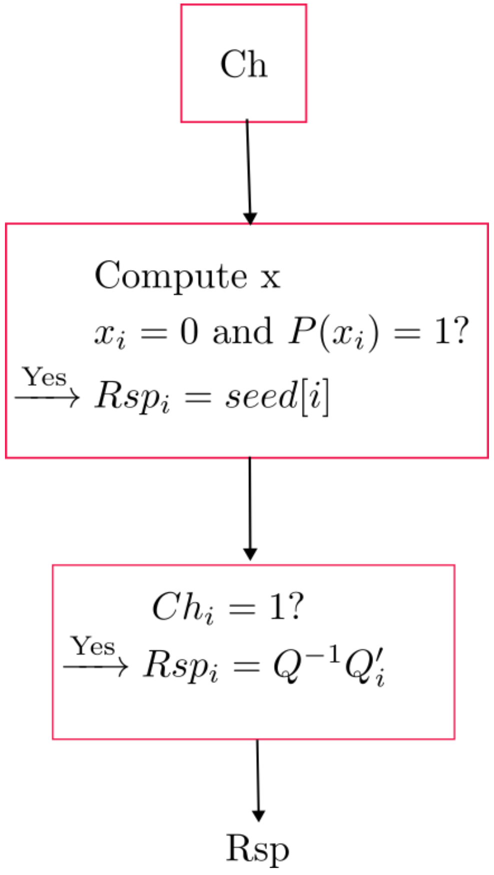
- So, we are getting both values seed[i'] (=Q'_i) and $Q^{-1}Q'_i$ for a non-zero Ch_i.

How to Prevent the Fault Attack?


Ch
Compute x
$x_i = 0$ and $P(x_i) = 1?$
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$
$Ch_i = 1?$
$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$
Rsp

- By injecting fault in the first part to get the value seed[i'] corresponding to Ch_i=1, which will not hamper the second part.
- So, we are getting both values seed[i'] (=Q'_i) and $Q^{-1}Q'_i$ for a non-zero Ch_i.
- The attack successfully done.

First Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1?$

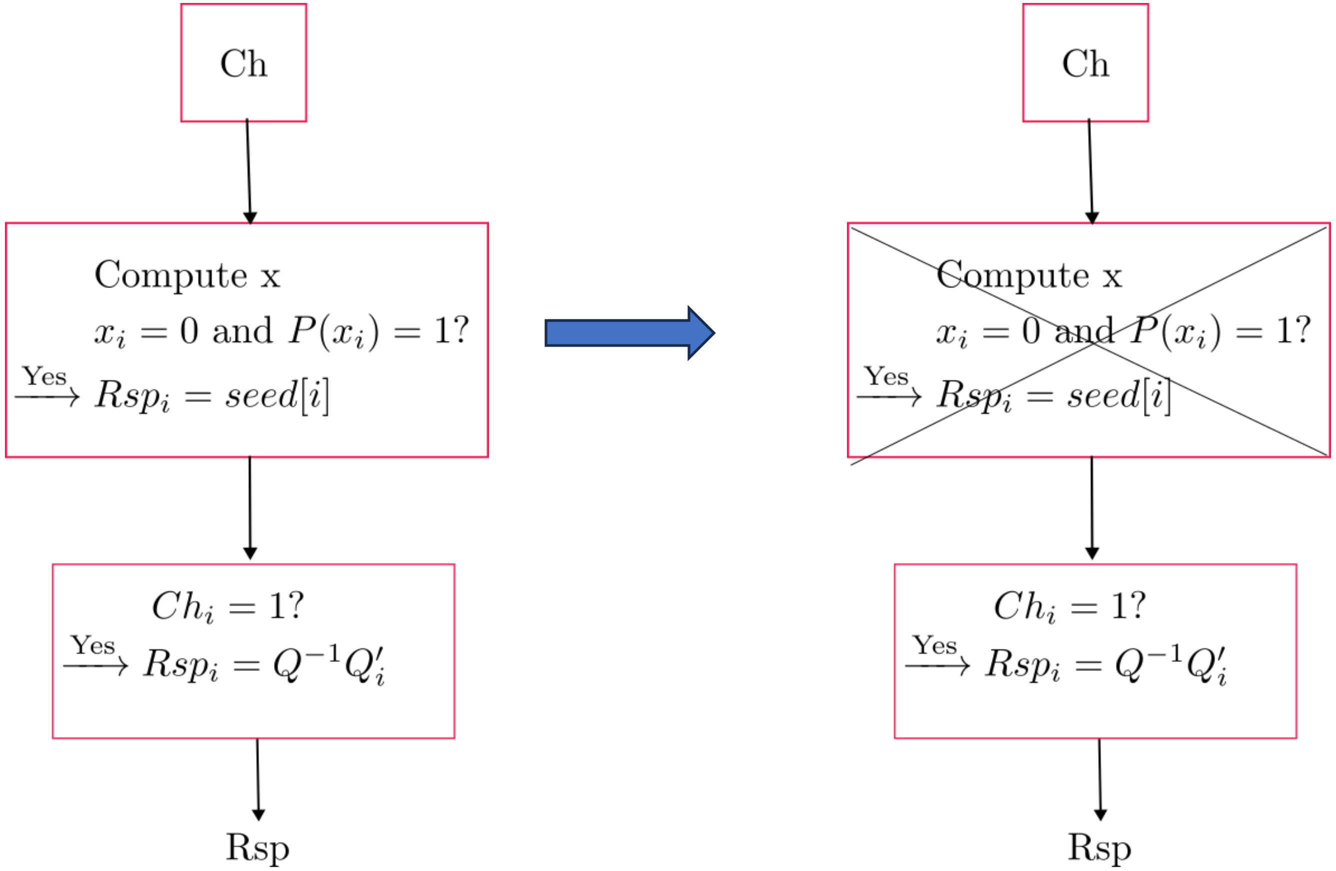
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

First Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

~~Compute x~~

~~$x_i = 0$ and $P(x_i) = 1?$~~

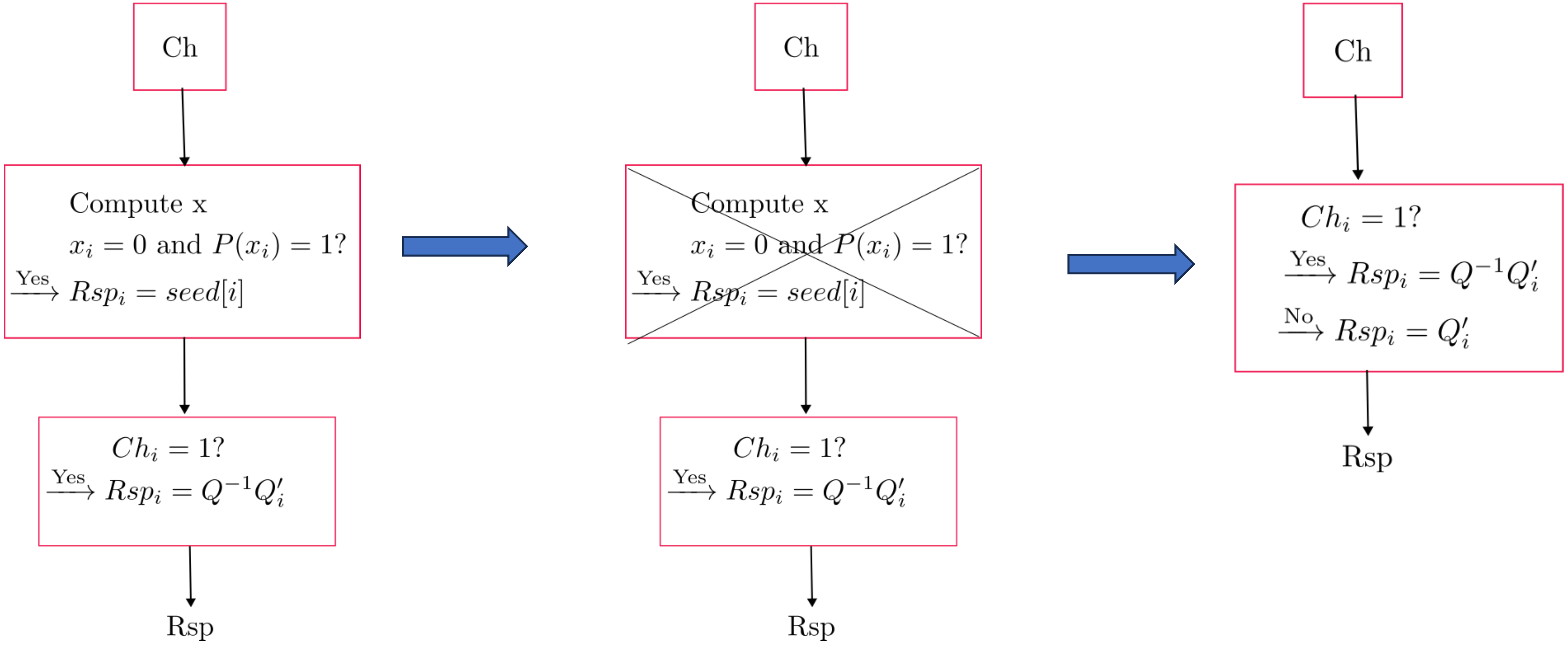
~~$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$~~

$Ch_i = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

First Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

~~Compute x~~

~~$x_i = 0$ and $P(x_i) = 1$?~~

~~$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$~~

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

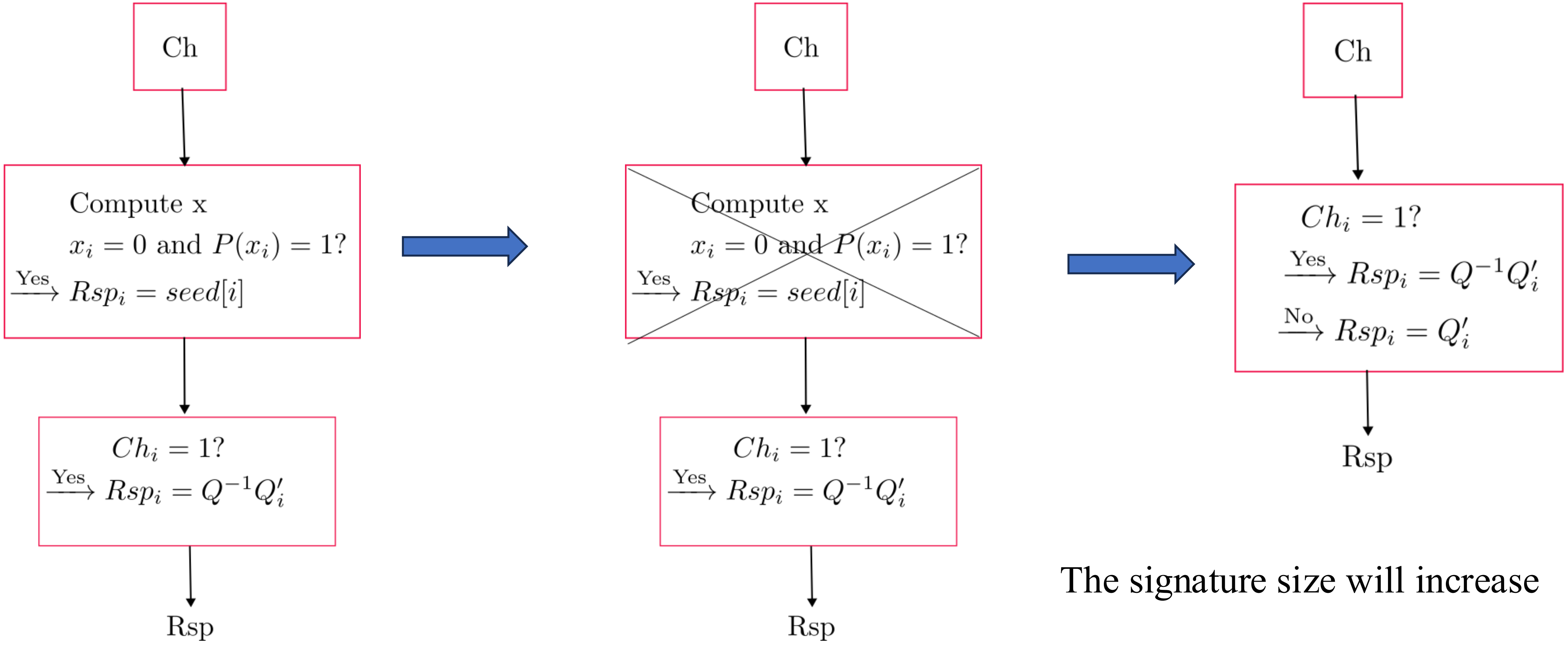
$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

$\xrightarrow{\text{No}} Rsp_i = Q'_i$

Rsp

First Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

~~Compute x~~

~~$x_i = 0$ and $P(x_i) = 1$?~~

~~$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$~~

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

$Ch_i = 1$?

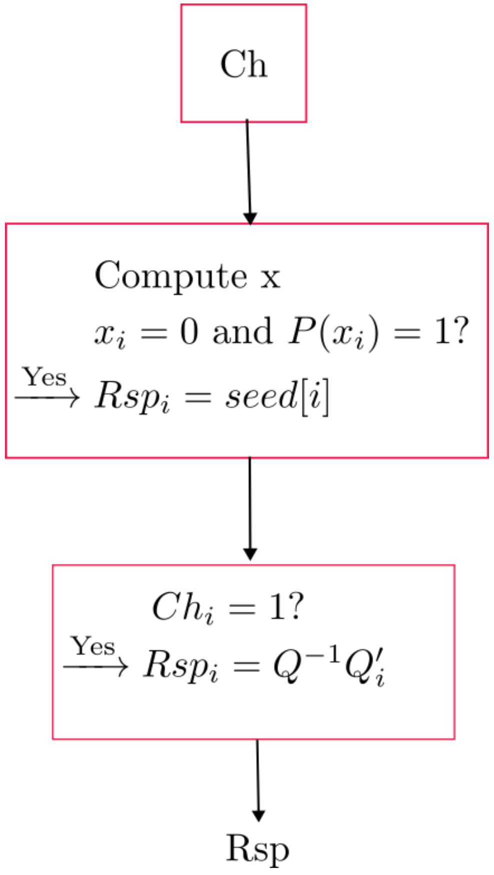
$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

$\xrightarrow{\text{No}} Rsp_i = Q'_i$

Rsp

The signature size will increase

Second Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1$?

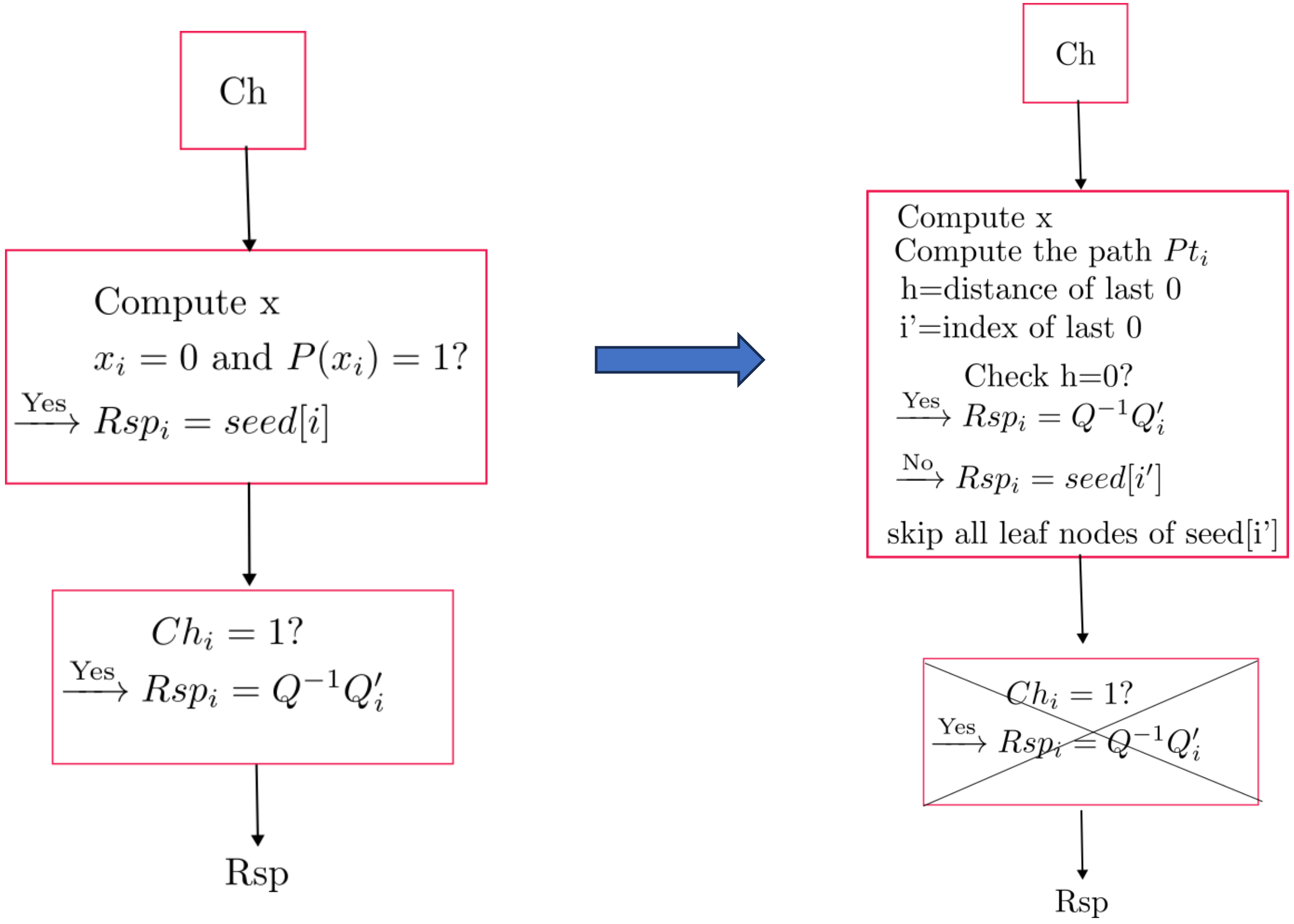
$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Second Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1?$

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

Compute x
Compute the path Pt_i
h=distance of last 0
i'=index of last 0

Check h=0?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

$\xrightarrow{\text{No}} Rsp_i = seed[i']$

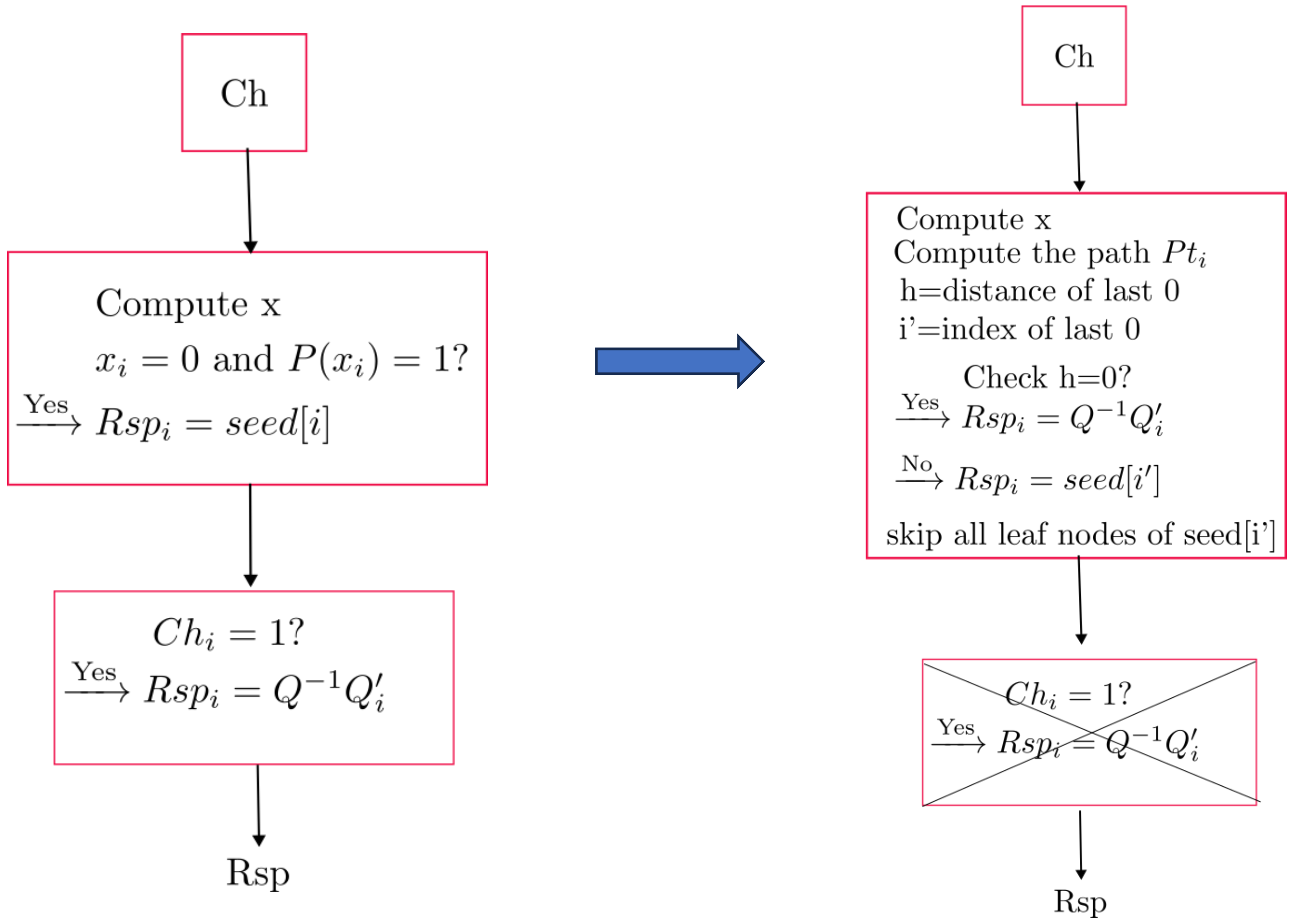
skip all leaf nodes of seed[i']

~~$Ch_i = 1?$~~

~~$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$~~

Rsp

Second Countermeasure



Ch

Compute x

$x_i = 0$ and $P(x_i) = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = seed[i]$

$Ch_i = 1$?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

Rsp

Ch

Compute x

Compute the path Pt_i

h=distance of last 0

i'=index of last 0

Check h=0?

$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$

$\xrightarrow{\text{No}} Rsp_i = seed[i']$

skip all leaf nodes of seed[i']

~~$Ch_i = 1$?~~

~~$\xrightarrow{\text{Yes}} Rsp_i = Q^{-1}Q'_i$~~

Rsp

- The signature size remain unchanged

Conclusion

- We show a fault attack on LESS and CROSS.

Conclusion

- We show a fault attack on LESS and CROSS.
- Our proposed fault detection method prevents erroneous secret recovery

Conclusion

- We show a fault attack on LESS and CROSS.
- Our proposed fault detection method prevents erroneous secret recovery
- It can be realized bit flip fault, struck at zero/one fault, instruction skip fault, etc.

Conclusion

- We show a fault attack on LESS and CROSS.
- Our proposed fault detection method prevents erroneous secret recovery
- It can be realized bit flip fault, struck at zero/one fault, instruction skip fault, etc.
- This attack can be applied to other schemes those uses the similar ZK structure
  - ✓E.g. MEDS

Conclusion

- We show a fault attack on LESS and CROSS.
- Our proposed fault detection method prevents erroneous secret recovery
- It can be realized bit flip fault, struck at zero/one fault, instruction skip fault, etc.
- This attack can be applied to other schemes those uses the similar ZK structure
  - ✓E.g. MEDS
- We have proposed two countermeasures that prevent the fault

Questions?

Thank you!